



SimCoder™

User's Guide

Powersim Inc.

SimCoder User's Guide

Version 2020a

Release 1

May 2020

Copyright © 2008-2020 Powersim Inc.

All rights reserved. No part of this manual may be photocopied or reproduced in any form or by any means without the written permission of Powersim Inc.

Disclaimer

Powersim Inc. ("Powersim") makes no representation or warranty with respect to the adequacy or accuracy of this documentation or the software which it describes. In no event will Powersim or its direct or indirect suppliers be liable for any damages whatsoever including, but not limited to, direct, indirect, incidental, or consequential damages of any character including, without limitation, loss of business profits, data, business information, or any and all other commercial damages or losses, or for any damages in excess of the list price for the licence to the software and documentation.

Powersim Inc.

Email: support@powersimtech.com

powersimtech.com

Contents

1 SimCoder Overview

- 1.1 Introduction 1
- 1.2 SimCoder Setup in Simulation Control 1
- 1.3 Elements for Code Generation 2

2 Code Generation - A Step-by-Step Approach

- 2.1 Overview 4
- 2.2 System in Continuous Domain 4
- 2.3 System in Discrete Domain 4
- 2.4 System Code Generation for Hardware Target 6
- 2.5 Subcircuit Code Generation 9
 - 2.5.1 Subcircuit Code Generation for Simulation 9
 - 2.5.2 Subcircuit Code Generation for Hardware Target 12
 - 2.5.3 Restrictions for Subcircuit Code Generation 13
- 2.6 System with Event Control 14

3 Event Handling

- 3.1 Basic Concept 16
- 3.2 Elements for Event Handling 16
- 3.3 Restrictions on Subcircuits with Events 17

4 SimCoder Libraries

- 4.1 Overview 19
- 4.2 Elements from Standard PSIM Library 19
 - 4.2.1 Defining Global Parameters in Parameter File 19
 - 4.2.2 Generating Sawtooth Waveform 21
- 4.3 Event Control Elements 21
 - 4.3.1 Input Event 21
 - 4.3.2 Output Event 22
 - 4.3.3 Default Event 23
 - 4.3.4 Event Connection 23
 - 4.3.5 Flag for Event Block First Entry 23
- 4.4 Global Variable 23
- 4.5 Interrupt 25
- 4.6 SimCoder C Block 26

5 IQmath Library

- 5.1 Overview 29
- 5.2 IQmath Data Type and Range/Resolution 29

6 F2833x Hardware Target

- 6.1 Overview 31
- 6.2 Hardware Configuration 33
- 6.3 DSP Clock 34
- 6.4 PWM Generators 34
 - 6.4.1 3-Phase PWM 35
 - 6.4.2 1-Phase PWM and 1-Phase PWM (phase shift) 37
 - 6.4.3 2-Phase PWM 41

6.4.4	Single PWM (shared with capture)	44
6.4.5	Synchronization Between PWM Blocks	44
6.5	Variable Frequency PWM	45
6.6	Start PWM and Stop PWM	46
6.7	Trip-Zone and Trip-Zone State	46
6.8	A/D Converter	47
6.9	Digital Input and Digital Output	51
6.10	Up/Down Counter	52
6.11	Encoder and Encoder State	52
6.12	Capture and Capture State	54
6.13	Serial Communication Interface (SCI)	55
6.13.1	SCI Configuration	55
6.13.2	SCI Input	56
6.13.3	SCI Output	56
6.14	Serial Peripheral Interface (SPI)	56
6.14.1	SPI Configuration	57
6.14.2	SPI Device	57
6.14.3	SPI Input	59
6.14.4	SPI Output	60
6.15	Controller Area Network (CAN) Bus	61
6.15.1	CAN Configuration	61
6.15.2	CAN Input	62
6.15.3	CAN Output	63
6.16	Interrupt Time	64
6.17	Project Settings and Memory Allocation	64

7 F2803x Hardware Target

7.1	Overview	67
7.2	Hardware Configuration	69
7.3	DSP Clock	69
7.4	PWM Generators	70
7.4.1	3-Phase PWM	71
7.4.2	1-Phase PWM and 1-Phase PWM (phase shift)	74
7.4.3	2-Phase PWM	80
7.4.4	Single PWM (shared with capture)	85
7.4.5	Synchronization Between PWM Blocks	85
7.5	Variable Frequency PWM	86
7.6	Start PWM and Stop PWM	87
7.7	Trip-Zone and Trip-Zone State	87
7.8	A/D Converter	88
7.9	Comparator	92
7.9.1	Comparator Input	92
7.9.2	Comparator Output	93
7.9.3	Comparator DAC	94
7.10	Digital Input and Digital Output	94
7.11	Up/Down Counter	95
7.12	Encoder and Encoder State	95
7.13	Capture and Capture State	96
7.14	Serial Communication Interface (SCI)	97
7.14.1	SCI Configuration	97
7.14.2	SCI Input	98
7.14.3	SCI Output	98

7.15	Serial Peripheral Interface (SPI)	99
7.15.1	SPI Configuration	99
7.15.2	SPI Device	100
7.15.3	SPI Input	101
7.15.4	SPI Output	102
7.16	Controller Area Network (CAN) Bus	104
7.16.1	CAN Configuration	104
7.16.2	CAN Input	105
7.16.3	CAN Output	106
7.17	Interrupt Time	107
7.18	Project Settings and Memory Allocation	107

8 F2806x Hardware Target

8.1	Overview	109
8.2	Hardware Configuration	111
8.3	DSP Clock	111
8.4	PWM Generators	112
8.4.1	3-Phase PWM	113
8.4.2	1-Phase PWM and 1-Phase PWM (phase shift)	116
8.4.3	2-Phase PWM Generator	122
8.4.4	Single PWM (shared with capture)	127
8.4.5	Synchronization Between PWM Blocks	127
8.5	Variable Frequency PWM	128
8.6	Start PWM and Stop PWM	129
8.7	Trip-Zone and Trip-Zone State	129
8.8	A/D Converter	130
8.9	Comparator	134
8.9.1	Comparator Input	134
8.9.2	Comparator Output	135
8.9.3	Comparator DAC	135
8.10	Digital Input and Digital Output	136
8.11	Up/Down Counter	137
8.12	Encoder and Encoder State	137
8.13	Capture and Capture State	138
8.14	Serial Communication Interface (SCI)	139
8.14.1	SCI Configuration	139
8.14.2	SCI Input	140
8.14.3	SCI Output	140
8.15	Serial Peripheral Interface (SPI)	141
8.15.1	SPI Configuration	141
8.15.2		141
8.15.3	SPI Input	143
8.15.4	SPI Output	144
8.16	Controller Area Network (CAN) Bus	146
8.16.1	CAN Configuration	146
8.16.2	CAN Input	147
8.16.3		147
8.17	Interrupt Time	149
8.18	Project Settings and Memory Allocation	149

9 F2802x Hardware Target

9.1	Overview	151
-----	----------	-----

9.2	Hardware Configuration	153
9.3	DSP Clock	153
9.4	PWM Generators	154
9.4.1	3-Phase PWM	155
9.4.2	1-Phase PWM and 1-Phase PWM (phase shift)	158
9.4.3	2-Phase PWM	164
9.4.4	Single PWM (shared with capture)	169
9.4.5	Synchronization Between PWM Blocks	169
9.5	Variable Frequency PWM	170
9.6	Start PWM and Stop PWM	171
9.7	Trip-Zone and Trip-Zone State	171
9.8	A/D Converter	172
9.9	Comparator	176
9.9.1	Comparator Input	176
9.9.2	Comparator Output	177
9.9.3	Comparator DAC	178
9.10	Digital Input and Digital Output	178
9.11	Capture and Capture State	179
9.12	Serial Communication Interface (SCI)	179
9.12.1	SCI Configuration	180
9.12.2	SCI Input	180
9.12.3	SCI Output	181
9.13	Serial Peripheral Interface (SPI)	181
9.13.1	SPI Configuration	181
9.13.2	SPI Device	182
9.13.3	SPI Input	184
9.13.4	SPI Output	185
9.14	Interrupt Time	186
9.15	Project Settings and Memory Allocation	186

10 F2837x Hardware Target

10.1	Overview	189
10.2	Hardware Configuration	191
10.3	DSP Clock	192
10.4	PWM Generators	192
10.4.1	3-Phase PWM, 1-phase PWM, and 1-phase PWM (phase shift)	193
10.4.2	2-Phase PWM Generator	198
10.4.3	Single PWM (shared with capture)	203
10.4.4	Synchronization Between PWM Blocks	204
10.5	Variable Frequency PWM	204
10.6	Start PWM and Stop PWM	205
10.7	PWM Trip-Zone And Trip-Zone State	205
10.7.1	PWM Trip-Zone	206
10.7.2	Trip-Zone State	209
10.8	ADC Voltage Reference	209
10.9	A/D Converter	210
10.10	Comparator	211
10.10.1	Comparator Input	211
10.10.2	Comparator DAC	213
10.11	D/A Converter	215
10.12	Digital Input and Digital Output	216
10.13	Digital Input Sample Time	217

10.14	Input X-BAR, Output X-BAR, and PWM X-BAR	217
10.14.1	Input X-BAR	217
10.14.2	Output X-BAR	218
10.14.3	PWM X-BAR	219
10.15	Encoder, Encoder State, and Encoder Index/Strobe Position	219
10.16	Up/Down Counter	221
10.17	Capture and Capture State	221
10.18	Serial Communication Interface (SCI)	222
10.18.1	SCI Configuration	223
10.18.2	SCI Input	223
10.18.3	SCI Output	223
10.19	Serial Peripheral Interface (SPI)	224
10.19.1	SPI Configuration	224
10.19.2	SPI Device	224
10.19.3	SPI Input	226
10.19.4	SPI Output	227
10.20	Controller Area Network (CAN) Bus	228
10.20.1	CAN Configuration	228
10.20.2	CAN Input	229
10.20.3	CAN Output	229
10.20.4	CAN Remote Request	230
10.21	Interrupt Time	231
10.22	Project Settings and Memory Allocation	231

11 F28004x Hardware Target

11.1	Overview	233
11.2	Hardware Configuration	235
11.3	DSP Clock	236
11.4	PWM Generators	236
11.4.1	3-Phase PWM, 1-phase PWM, and 1-phase PWM (phase shift)	237
11.4.2	2-Phase PWM Generator	242
11.4.3	Single PWM (shared with capture)	247
11.4.4	Synchronization Between PWM Blocks	247
11.5	Variable Frequency PWM	248
11.6	Start PWM and Stop PWM	249
11.7	PWM Trip-Zone And Trip-Zone State	249
11.7.1	PWM Trip-Zone	250
11.7.2	Trip-Zone State	253
11.8	ADC Voltage Reference	253
11.9	A/D Converter	253
11.10	PGA and PGA Gain	256
11.11	Comparator	257
11.11.1	Comparator Input	257
11.11.2	Comparator DAC	259
11.12	D/A Converter	260
11.13	Digital Input, Output, and Sample Time	261
11.14	Input X-BAR, Output X-BAR, and PWM X-BAR	263
11.14.1	Input X-BAR	263
11.14.2	Output X-BAR	263
11.14.3	PWM X-BAR	264
11.15	Encoder, Encoder State, and Encoder Index/Strobe Position	265
11.16	Up/Down Counter	266

11.17	Capture and Capture State	267
11.18	Serial Communication Interface (SCI)	268
11.18.1	SCI Configuration	268
11.18.2	SCI Input	269
11.18.3	SCI Output	269
11.19	Serial Peripheral Interface (SPI)	269
11.19.1	SPI Configuration	270
11.19.2	SPI Device	270
11.19.3	SPI Input	272
11.19.4	SPI Output	273
11.20	Controller Area Network (CAN) Bus	274
11.20.1	CAN Configuration	274
11.20.2	CAN Input	275
11.20.3	CAN Output	275
11.20.4	CAN Remote Request	276
11.21	Interrupt Time	277
11.22	Project Settings and Memory Allocation	277

12 PE-Expert4 Hardware Target

12.1	Overview	279
12.2	DSP Board	279
12.2.1	Start and Stop PWM	279
12.2.2	LED Output	280
12.2.3	Timer Interrupt Priority	280
12.3	PEV Board	280
12.3.1	PEV Board Configuration	281
12.3.2	PWM Generators	281
12.3.3	A/D Converter	284
12.3.4	Digital Input / Capture / Counter	284
12.3.5	Digital Output	286
12.3.6	Encoder	286
12.4	PSPWM Board	287
12.4.1	PSPWM Board Configuration	287
12.4.2	PWM Generators (PSPWM-24, PSPWM-144, and EXGATE-18)	287
12.4.3	A/D Converter	292
12.4.4	Digital Input and Output	293
12.5	ADC Board	293
12.6	PE-Expert4 Runtime Library Functions	294

13 TI DMC Library

13.1	Overview	295
13.2	ACI_FE: Flux Estimator of 3-phase Induction Motors	296
13.3	ACI_SE: Speed Estimator of 3-phase Induction Motors	297
13.4	ANGLE_MATH: Angle Wrap	299
13.5	CLARKE: Clarke Transformation	299
13.6	CUR_MOD: Current Model	301
13.7	IMPULSE: Impulse Generator	302
13.8	IPARK: Inverse Park Transformation	303
13.9	PARK: Park Transformation	304
13.10	PHASE_VOLT: Phase Voltage Reconstruction	305
13.11	PID Controllers	305
13.11.1	PI: PI Controller with Anti-Windup	306

13.11.2 PI_REG4: PI Controller with Anti-Windup	307
13.11.3 PI_POS: PI Controller with Position Error Wrapper	307
13.11.4 PI_POS_REG4: PI Controller with Position Error Wrapper	308
13.11.5 PID_REG3: PID Controller with Anti-Windup	309
13.11.6 PID_GRANDO: PID Controller	309
13.12 RAMP_GEN: Ramp Generator	311
13.13 Ramp Control	312
13.13.1 RMP_CNTL: Ramp Control	312
13.13.2 RMP2CNTL: Ramp 2 Control	313
13.13.3 RMP3CNTL: Ramp 3 Control	314
13.14 SOMPOS: Sliding-Mode Rotor Position Observer	314
13.15 Speed Calculators	316
13.15.1 SPEED_EST: Speed Calculator	316
13.15.2 SPEED_FR: Speed Calculator with QEP Sensor	317
13.15.3 SPEED_PRD: Speed Calculator with Period Measurement	317
13.16 Space Vector Generators	318
13.17 VHZ_PROFILE: Volt/Hertz Profile for AC Induction Motors	319

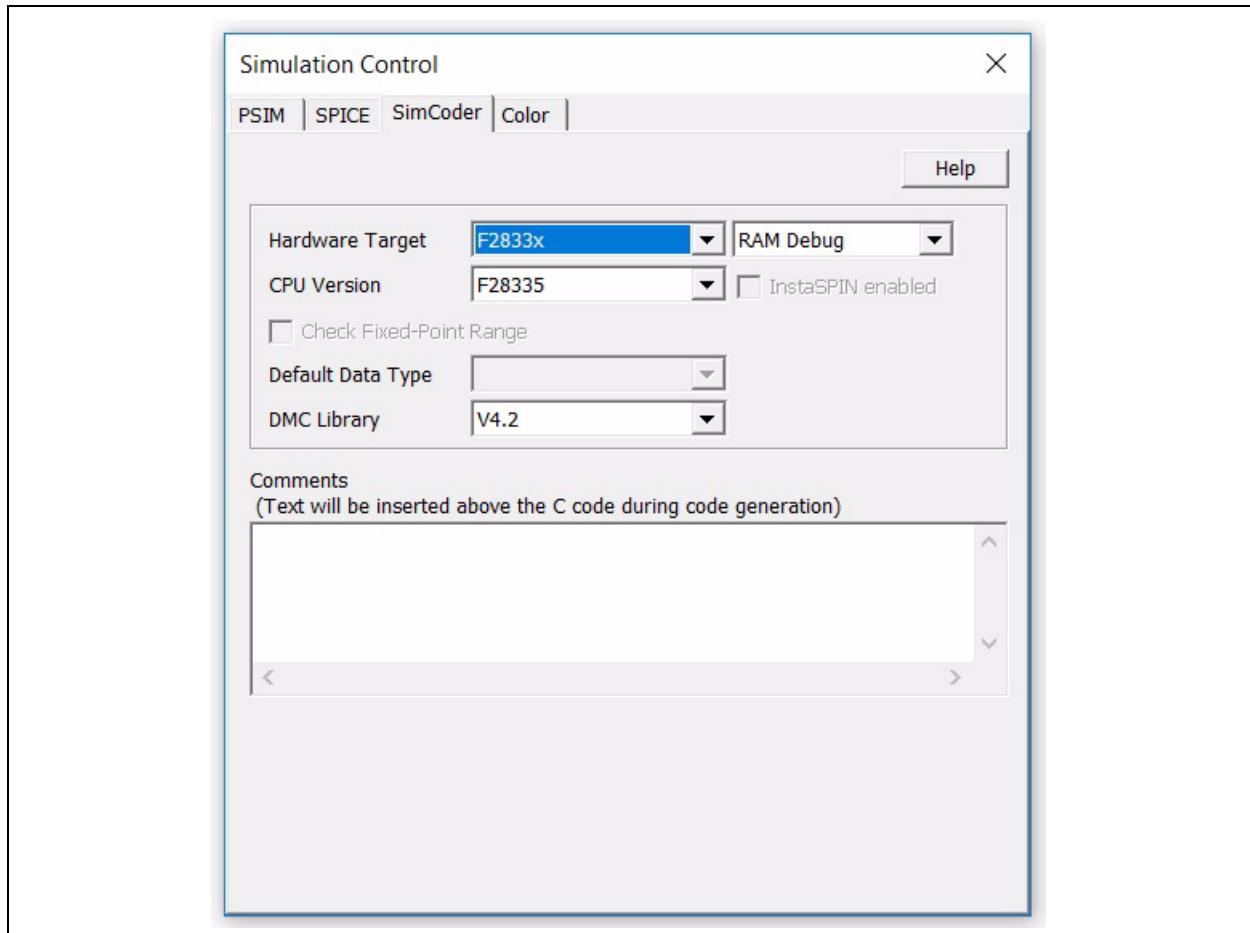
SimCoder Overview

1.1 Introduction

SimCoder¹ is an add-on option of the PSIM software. It generates C code from PSIM schematics. With specific hardware target libraries, the C code generated by SimCoder can run directly on target hardware platforms. This manual describes how to use SimCoder.

1.2 SimCoder Setup in Simulation Control

SimCoder setup is done in the SimCoder tab in the **Simulation Control** block, as shown below.



One must select and set these parameters properly in order for SimCoder to generate code correctly. The setup process is explained below.

Hardware Targets:

SimCoder supports the hardware targets of Texas Instruments' C2000TM real-time control MCUs. The options of "None" in the pulldown list of Hardware Target is simulation only.

For each F28xxx Target, the supported CPU versions in the product series are listed below. Each CPU version

1. SimCoderTM is a trademark of Powersim Inc., and is copyright by Powersim Inc., 2008-2020

has the choice of the number of pins for the choice of IC package.

Target	SimCoder Supported CPU Version
F2833x	28335, 28334, 28332.
F2803x	28035, 28034, 28033, 28032, 28031, 28030.
F2806x	28069, 28068, 28067, 28066, 28065, 28064, 28063, 28062.
F2802x	28027, 28026, 28023, 28022, 28021, 28020, and 280200.
F2837x	28374S/D, 28375S/D, 28376S/D, 28377S/D, 28379S/D.
F28004x	280049, 280049C, 280048, 280048C, 280045, 280041, 280041C, 280040, 280040C.
PE-Expert4	For the PE-Expert4 DSP hardware. PE-Expert4 is a DSP development platform produced by Myway Co. It uses TI's floating-point DSP TMS320C6657 and Myway's PE-OS library.

Project Configuration

For *F28xxx* Targets, the project configuration can be set as *RAM Debug*, *RAM Release*, *Flash Release*, or *Flash RAM Release*. For *PE-Expert4* Target, the project configuration can be set as *PE-ViewX*.

Check Fixed-Point Range:

This is for the fixed-point targets (*F2803x/F2806x/F2802x* Targets) only. If the box is checked, the SimCoder will check the data in the simulation and provide a list of data range. In that list, the data which are near or over the range will be highlighted.

Default Data Type

When the hardware target is of floating-point type, this section is automatically chosen. If the hardware target is of fixed-point type or *None*, one must select one of the available default data types in the pull-down menu.

- For fixed-point targets, (*F2803x/F2806x/F2802x* Targets), it can be one in the pull-down list: Integer, IQ1, IQ2, .., IQ30.
- When there is no target, it can be one in the pull-down list: Float, Integer, IQ1, IQ2,..., IQ30

DMC Library Version

Texas Instruments's DMC (Digital Motor Control) library is composed of C functions (or macros) developed for C2000 DSP devices for motor control users.

To take advantage of this resource, SimCoder integrated the DMC library functions into the PSIM element library for code generation.

Some of these macros have different versions released by TI. SimCoder supports versions V4.0, V4.1, and V4.2. Please note that, once a specific version is selected, other versions' macros are disabled.

Comments

The Comments area at the bottom of the SimCoder tab allows users to add comments to the code generated by SimCoder. All text in this area will be added as comments to the beginning of the C code.

1.3 Elements for Code Generation

All the elements under the menu **Elements >> Event Control** and **Elements >> SimCoder** are for code generation.

Elements for each type of hardware targets can be found under the menu **Elements >> SimCoder**, under its own sub-menu respectively.

Many elements in the standard PSIM library can also be used for code generation. In order to differentiate the elements in the standard library that can be used for code generation from the ones that can not, under **Options >> Settings >> Advanced**, if the option box "Show image next to elements that can be used for code

generation" is checked, a small image  will appear next to these elements that can be used for code generation.

Also, when this option box is checked, the image of  will appear next to each of the elements for F2833x, F2803x, F2806x, and F2802x Targets, and  for PE-Expert4 Target.

For a list of elements in the standard library that can be used for code generation, please refer to Section 4.2.

Code Generation - A Step-by-Step Approach

2.1 Overview

In general, automatic code generation using SimCoder involves the following steps:

- Design and simulate a system in PSIM with the control in continuous domain.
- Convert the control section of the system into discrete domain and simulate the system.
- If there is no hardware target, place the control section in a subcircuit, and generate the code.
- If there is a hardware target, modify the system by including hardware elements, and run the simulation to validate the results. Then generate the code.

The first two steps, however, are not mandatory. One could, for example, create a schematic in PSIM and generate the code directly without simulating the system.

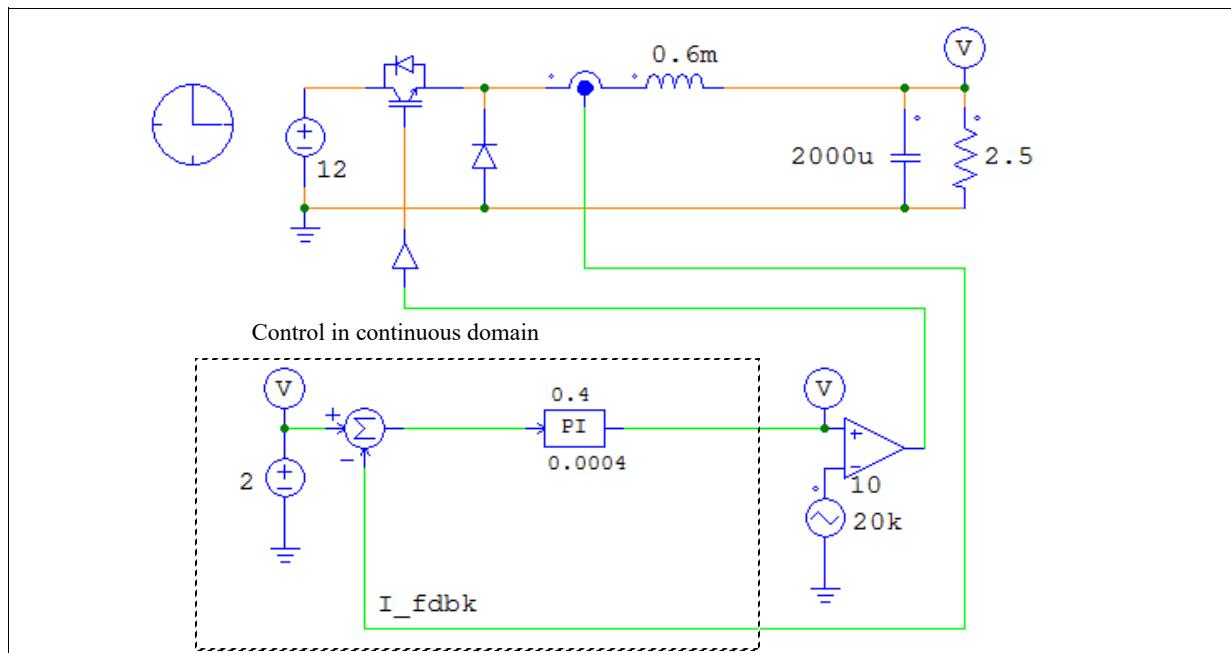
Please note that code can only be generated when control is in discrete domain, not in continuous domain. Therefore, Digital control Module is needed for SimCoder.

Simple examples are used in the sections below to illustrate the code generation process.

2.2 System in Continuous Domain

Often a system is designed and simulated in continuous domain first. Below is a simple dc converter circuit with current feedback. The PI (proportional-integral) controller in the control circuit is designed in the continuous s-domain. The PI gain k and time constant T are: $k = 0.4$, and $T = 0.0004$. The switching frequency is 20 kHz.

The objective of this exercise is to generate the C code for the control circuit in the dotted box. To perform the code generation, the first step is to convert the analog PI controller in s-domain to the digital PI controller in discrete z-domain.

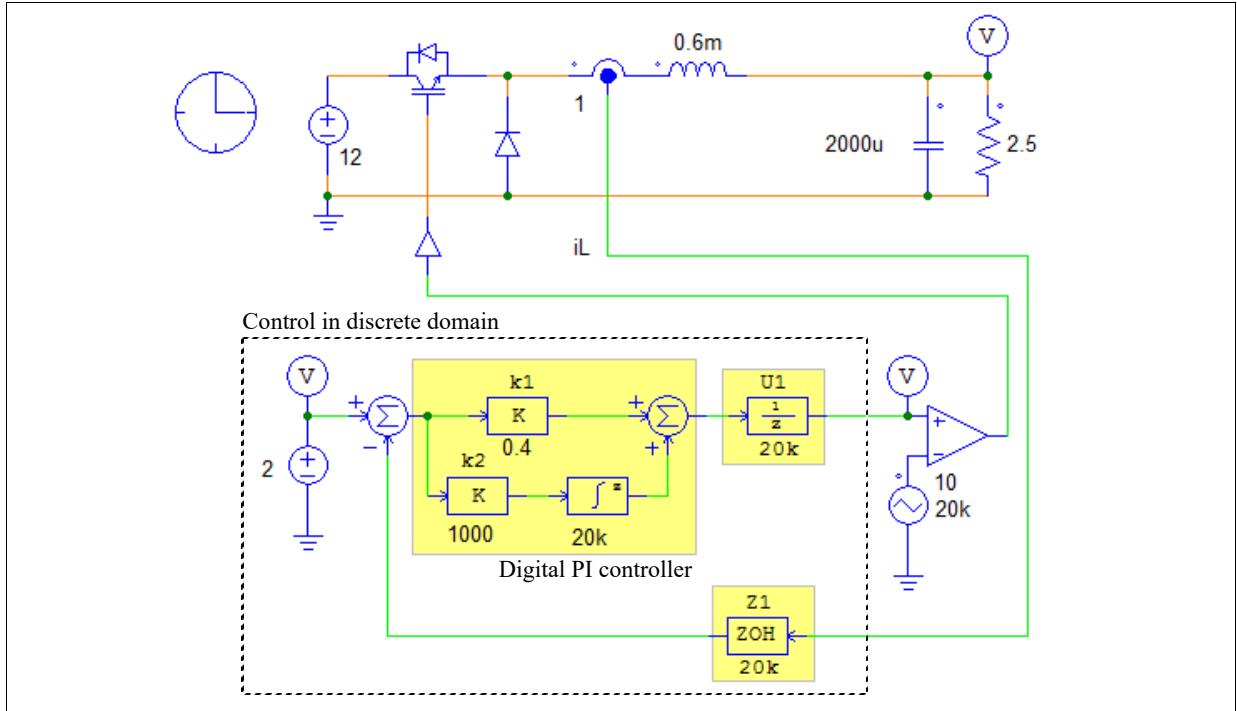


2.3 System in Discrete Domain

To convert an analog controller into a digital controller, one can use the *s2z Converter* program that comes with the Digital Control Module. To launch the program, in PSIM, choose **Utilities > s2z Converter**.

Different conversion methods can be used to convert an analog controller to a digital controller. The most commonly used methods are Bilinear (also called Tustin or Trapezoidal) method and Backward Euler method. In this example, we will use the Backward Euler method. With the sampling frequency the same as the switching frequency of 20 kHz, we will convert the analog PI controller to the digital PI controller. From the conversion program, we have the digital PI controller parameters as: $k_1 = 0.4$ for the proportional portion and $k_2 = 1000$ for the integral portion.

The circuit with the digital controller is shown below:



As compared to the control circuit in continuous domain, there are three changes in this circuit, as highlighted by the yellow boxes. First, the analog PI controller is replaced by the digital PI controller. The "Algorithm Flag" of the digital integrator is set to 1 (for Backward Euler method), and the sampling frequency is set to 20 kHz. The gains k_1 and k_2 are obtained from the conversion program as described above.

In addition, a zero-order-hold block Z_1 is used to simulate the A/D converter in digital hardware implementation for sampling the feedback current i_L . A unit-delay block U_1 is used to model the one-cycle delay inherent in digital control implementation. The delay is due to the fact that, usually quantities are sampled at the beginning of a cycle, and controller parameters are calculated within the cycle. But since it takes time to perform the calculation, the newly calculated quantities are normally not used until the beginning of the next cycle.

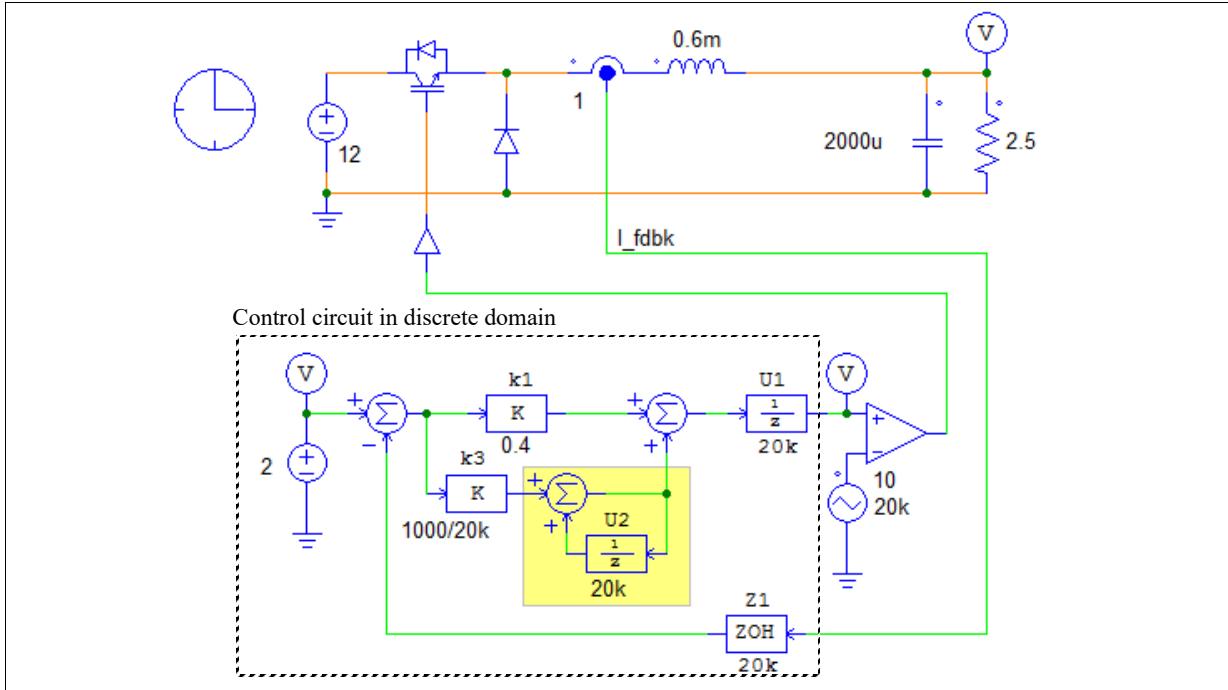
Note that the converted digital controller should result in a stable control loop and desired performance. If the simulation results with the digital control are not stable or not as desired, one needs to go back to the analog control system, re-design the analog controller, and repeat the process.

With the Backward Euler method, we can also represent the output-input relationship in the time domain as follows:

$$y(n) = y(n-1) + T_s * u(n)$$

where $y(n)$ and $u(n)$ are the output and input at the current time, $y(n-1)$ is the output at the previous sampling period, and T_s is the sampling period. Using the equation above, we can replace the discrete integrator in the

above circuit with a summer and a unit-delay block, as shown below:



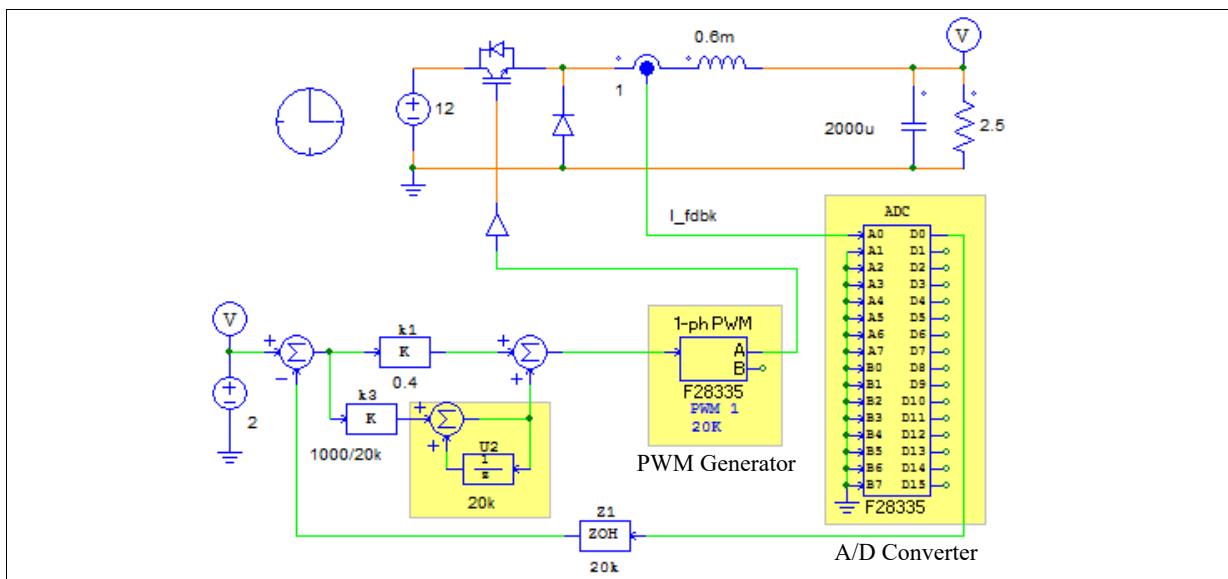
Note that, due to the factor T_s in the equation, the gain of the proportional block k_2 needs to be divided by the sampling frequency of 20kHz. The advantage of this circuit is that it is easier to start or stop the integration of the integrator.

With the control circuit in discrete domain, one is now ready to move on to the next step.

2.4 System Code Generation for Hardware Target

To generate system code for specific hardware target, the circuit schematic must be modified to include relevant hardware elements. Also, variables must be scaled properly to take into account the valid ranges of hardware elements.

Below is the same example circuit but with F2833x Target hardware elements added. For better illustration, the hardware elements are highlighted in yellow.



In the example circuit schematic, the following changes are made:

- An A/D converter is inserted between the current sensor and the control subcircuit. Special attention must be paid to the input range of the A/D converter. If the current sensor output is out of the range of the A/D converter, it must be scaled back accordingly. For this particular example, the A/D converter settings are as shown below.

- **ADC Mode:** *Start-stop (8-channel)*. This means the A/D conversion will be triggered by PWM generator, and only half of the ADC converter is used.
- **Ch A0 Mode:** *DC*. This set the input signal range from 0 to +3V
- **Ch A0 Gain:** *1.0*.

- The comparator and the carrier wave are replaced with the hardware PWM generator. The PWM generator settings relevant to this example are:

- **PWM Source:** *PWM1*. This defines the PWM module in F28335 processor.
- **Output Mode:** *Use PWM A*. This defines the PWM output port
- **Sampling Frequency:** *20k*. This defines the sampling frequency at 20 kHz.
- **Carrier Wave Type:** *Sawtooth (start high)*. This setting will be explained further in Chapter 5.
- **Trigger ADC:** *Trigger ADC Group A*. This setting will be explained further in Chapter 5.
- **ADC Trigger Position:** *0*. This setting will be explained further in Chapter 5.
- **Peak-to-Peak Value:** *10*. This defines the range of the input signal to this PWM generator.

- The unit delay block U1 is removed. This is because the PWM generator contains one sampling period delay inherently,

- In **Simulation Control**, go to **Parameters** tab, in **SimCoder** section:

- **Hardware Type** is set to *F2833x*, with *RAM Debug*.
- **CPU Version** is set to *F28335*
- **Default Fixed-Point Position** is not applicable because it is set to **Float**

- User can add a comment section to the beginning of the generated code. To create and edit the comments, in **Simulation Control**, go to the **SimCoder** tab, and enter or edit the comments.

To check the validity of the changes after the hardware elements are added, run simulation for this system. The results of this system should be very close to the results of the system with the digital control in Section 2.2.

Once the simulation results are verified, C code can be generated by clicking **Simulate >> Generate Code**. The code generated for F2833x hardware is ready to run without any changes.

Below is the C code generated by the SimCoder for the system described above.

```
*****
// This code is created by SimCoder Version 9.3.3 for TI F2833x Hardware Target
//
// SimCoder is copyright by Powersim Inc., 2009-2013
//
// Date: February 24, 2014 14:36:33
*****  

#include<math.h>
#include"PS_bios.h"
typedef float DefaultType;
#defineGetCurTime() PS_GetSysTimer()  
  

interrupt void Task();  
  

DefaultTypefGbliref = 0;
DefaultTypefGblU2 = 0;  
  

interrupt void Task()
{
    DefaultTypefU2, fSUMP1, fSUMP3, fk3, fk1, fSUM1, fz1, fTI_ADC1, fVDC2;  
  

    PS_EnableIntr();
    fU2 = fGblU2;  
  

    fTI_ADC1 = PS_GetDcAdc(0);
    fVDC2 = 2;
```

```

fZ1 = fTI_ADC1;
fSUM1 = fVDC2 - fZ1;
fk1 = fSUM1 * 0.4;
fk3 = fSUM1 * (1000.0/20000);
fSUMP3 = fk3 + fU2;
fSUMP1 = fk1 + fSUMP3;
PS_SetPwm1RateSH(fSUMP1);

#ifndef_DEBUG
    fGbliref = fVDC2;
#endif
    fGblU2 = fSUMP3;
    PS_ExitPwm1General();
}

void Initialize(void)
{
    PS_SysInit(30, 10);
    PS_StartStopPwmClock(0);
    PS_InitTimer(0, 0xffffffff);
    PS_InitPwm(1, 0, 20000*1, (4e-6)*1e6, PWM_POSI_ONLY, 42822); // pwnNo, waveType, frequency, deadtime,
outtype
    PS_SetPwmPeakOffset(1, 10, 0, 1.0/10);
    PS_SetPwmIntrType(1, ePwmIntrAdc0, 1, 0);
    PS_SetPwmVector(1, ePwmIntrAdc0, Task);
    PS_SetPwmTzAct(1, eTZHighImpedance);
    PS_SetPwm1RateSH(0);
    PS_StartPwm(1);

    PS_ResetAdcConvSeq();
    PS_SetAdcConvSeq(eAdc0Intr, 0, 1.0);
    PS_AdcInit(1, !1);

    PS_StartStopPwmClock(1);
}

void main()
{
    Initialize();
    PS_EnableIntr(); // Enable Global interrupt INTM
    PS_EnableDbgm();
    for (;;) {
    }
}

```

The generated code has the following structure:

Interrupt void Task (): The interrupt service routine for 20 kHz. It is called in every 20-kHz cycle.

void Initialize (): The initialization routine. It initializes hardware.

void main (): The main program. It calls the initialization routine, and runs an infinite loop.

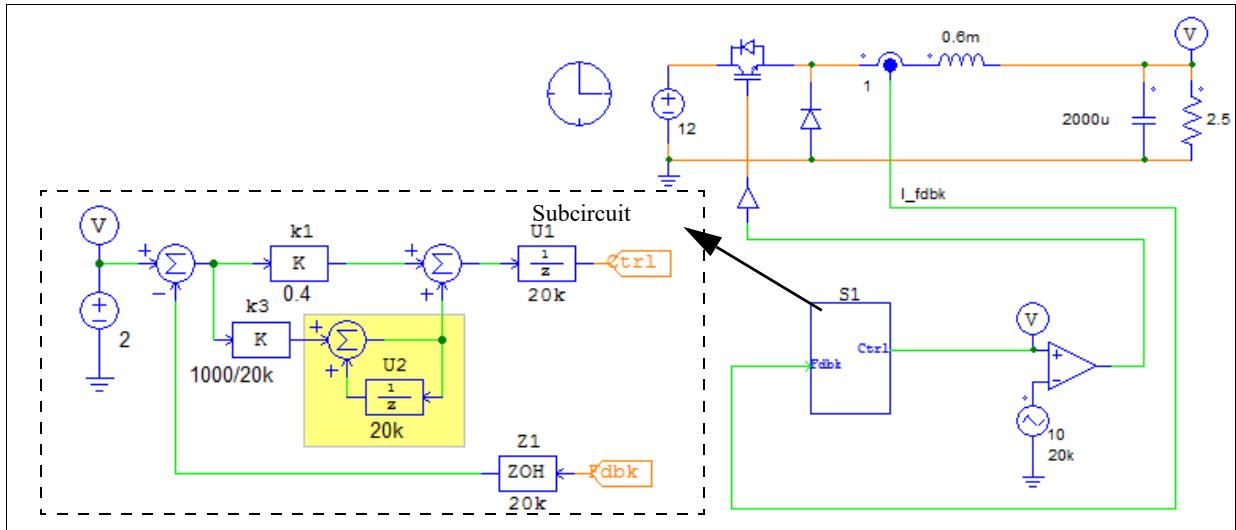
Note that in this example, all the control blocks run at the 20-kHz sampling rate. If there were blocks that run at a different sampling rate, another service routine would be created. One interrupt service routine will correspond to one sampling rate. For blocks that do not have sampling rates associated with them, the corresponding code will be placed in the main program.

This code and all necessary project files are stored in a sub-folder in the same directory as the main schematic file. User can load the project files into TI's Code Composer Studio environment, compile the code, and upload it onto the DSP hardware for real-time operation.

2.5 Subcircuit Code Generation

In PSIM, a section of the circuit can form a subcircuit. SimCoder can generate code for each subcircuit, given the condition that all the elements in the subcircuit are supported by code generation. There are some other restrictions which are listed in Section 2.5.3

To illustrate the code generation for a subcircuit, we continue to use the example system in Section 2.3. The control circuit within the dotted line is placed inside a subcircuit, as shown below.



To create the subcircuit, select the circuit in the dotted box. Right click the mouse to display the pull-down menu. From the menu, select **Create Subcircuit**, and define the subcircuit file name.

The subcircuit for code generation excluded the comparator and the carrier wave source. One of the reasons for this arrangement is, in most of the hardware setup, these two functions are either implemented by external hardware or embedded in microcontroller's peripheral interface. The other reason is, for simulation, the carrier wave and the comparator must be calculated at every time step, but the code is executed at the sampling rate of 20kHz. In SimCoder generated code, the sampling rate of every element must be defined. The comparator has two inputs: one is from the controller which has 20 kHz sampling rate, and the other is the carrier wave source which is undefined. In such cases, SimCoder will assume that both inputs of the comparator have the same sampling rate as the input which is defined.

SimCoder can generate code for a subcircuit either for simulation or for hardware target operation. These two types of code are not interchangeable. The subcircuit code generated for simulation can not be used in hardware target, and vice versa. These two situations are explained in the subsections below.

2.5.1 Subcircuit Code Generation for Simulation

To generate code for simulation, follow the steps below:

- While in the main circuit, right click the mouse at the subcircuit block and select **Attributes**.
- At the bottom of the Attributes dialog window, click the button **Generate Code**, and select the option **Generate Code for Simulation**.
- If wanted, user can add a comment section to the beginning of this code. To create and edit the comments, in **Simulation Control**, go to the **SimCoder** tab, and enter or edit the comments in the dialog window before clicking the **Generate Code** button.

The beginning of generated code is shown below.

```
*****
// This code is created by SimCoder Version 9.3.3
//
// SimCoder is copyright by Powersim Inc., 2009-2013
// Date: February 24, 2014 14:55:33
*****  

#include <stdio.h>
#include <math.h>
```

```

#define ANALOG_DIGIT_MID 0.5
#define INT_START_SAMPLING_RATE 1999999000L
#define NORM_START_SAMPLING_RATE 2000000000L

typedef void (*TimerIntFunc)(void);
typedef double DefaultType;
DefaultType *inAry = NULL, *outAry;
DefaultType *inTmErr = NULL, *outTmErr;

double fCurTime;
double GetCurTime() {return fCurTime;}

/* The input/output node definition for C/DLL block.
   The 2nd display node (outAry[1]): From element 'S1_ioref'.
*/
/* The C block for the generated code has the following additional output port(s):
   2 - S1.ioref
*/
void _SetVP6(int bRoutine, DefaultType fVal);
void InitInOutArray()
{... ...}

void FreeInOutArray()
{ ... ...}

void CopyInArray(double* in)
{ ... ...}

void CopyOutArray(double* out)
{ ... ...}

void Task();
void TaskS1(DefaultType fIn0, DefaultType *fOut0);

DefaultType fGblS1_U1 = 0;
DefaultType fGblS1_U2 = 0;

void TaskS1(DefaultType fIn0, DefaultType *fOut0)
{ ... ...}

void Task()
{
    TaskS1(inAry[0],&outAry[0]);
}

typedef struct {
    TimerIntFunc func;
    long sampRate;
    double tmLastIntr;
} TimeChk;
#define NUM_TIMER_INTR 1
TimeChk lGbl_TimeOverChk[NUM_TIMER_INTR] = {
    {Task, 20000, 0}};

void InitAllTaskPtr(void)
{
    lGbl_TimeOverChk[0].func = Task;
    lGbl_TimeOverChk[0].sampRate = 20000;
}

void _SetVP6(int bRoutine, DefaultType fVal)
{
    static DefaultType val = 0.0;
    if (bRoutine) {
        val = fVal;
    }
    outAry[1] = val;
}

```

At the end of the generated code from the subcircuit, it contains *SimulationStep* function, *SimulationBegin* function, and *SimulationEnd* function at the end, as shown below. These functions can be used in C Block which can replace the subcircuit.

```

void SimulationStep(
    double t, double delt, double *in, double *out,
    int *pnError, char * szErrorMsg,
    void ** reserved_UserData, int reserved_ThreadIndex, void * reserved_AppPtr)
{ ... ... }

void SimulationBegin(
    const char *szId, int nInputCount, int nOutputCount,
    int nParameterCount, const char ** pszParameters,
    int *pnError, char * szErrorMsg,
    void ** reserved_UserData, int reserved_ThreadIndex, void * reserved_AppPtr)
{
    InitInOutArray();
}

void SimulationEnd(const char *szId, void ** reserved_UserData, int reserved_ThreadIndex, void *
reserved_AppPtr)
{
    FreeInOutArray();
}

```

Replacing Subcircuit with C Block

In the subcircuit attribute dialog, there is a check box called **Replace subcircuit with generated code for simulation**. When this box is checked, the simulation will be done as if the system contains a C Block instead of a subcircuit. User do not need to go through the procedure to replace the subcircuit with a C block.

However, if the user prefer to modify the C-code at will, one can place the generated code in a C Block by following the steps below.

In the above example, the code generated from the subcircuit contains four sections: *SimulationStep*, *SimulationBegin*, *SimulationEnd* and the rest of the code. Similarly, the C Block is also composed of these four sections. A C Block contains the following sections:

```

#include <Stdlib.h>
#include <String.h>
#include <math.h>
#include <Psim.h>

// PLACE GLOBAL VARIABLES OR USER FUNCTIONS HERE...
... ...

///////////////////////////////
// FUNCTION: SimulationStep
{
// ENTER YOUR CODE HERE...

}

///////////////////////////////
// FUNCTION: SimulationBegin
{
// ENTER INITIALIZATION CODE HERE...

}

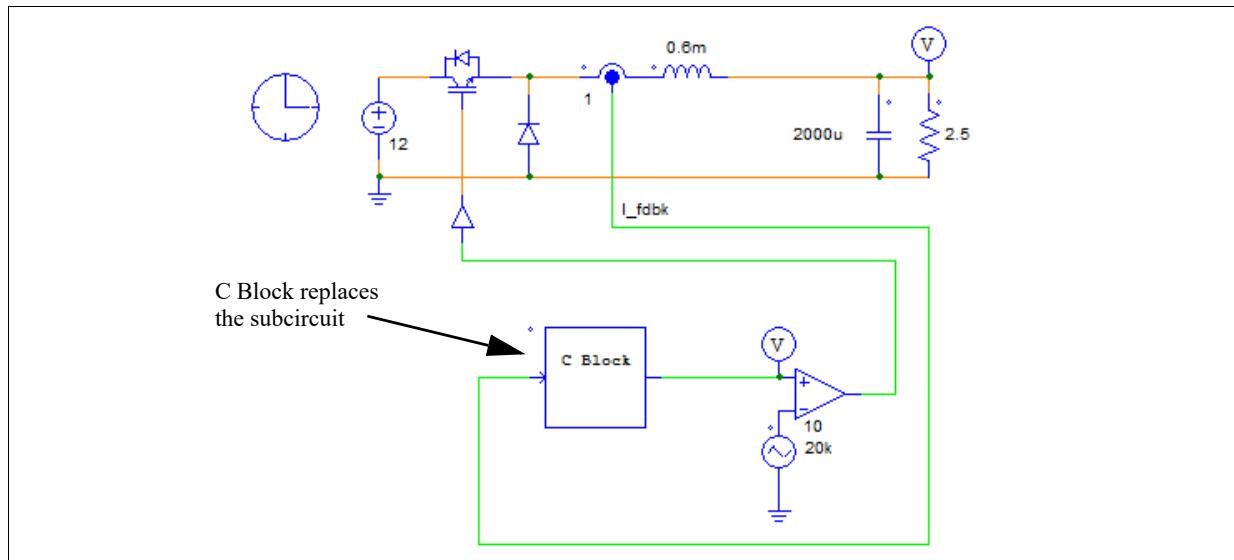
///////////////////////////////
// FUNCTION: SimulationEnd
{
}

```

To create a C Block in the main circuit, use the pull-down menu **Elements >> Other >> Function Blocks >> C Block**. Then, copy the each section of the generated code to each section of the C Block: from the *SimulationStep* function to *SimulationStep* section in the C Block, from the *SimulationBegin* function to

SimulationBegin section, from the *SimulationEnd* function to *SimulationEnd* section, and put the rest of the code to the *User Functions* section.

The example circuit with the subcircuit replaced with a C Block is shown below.

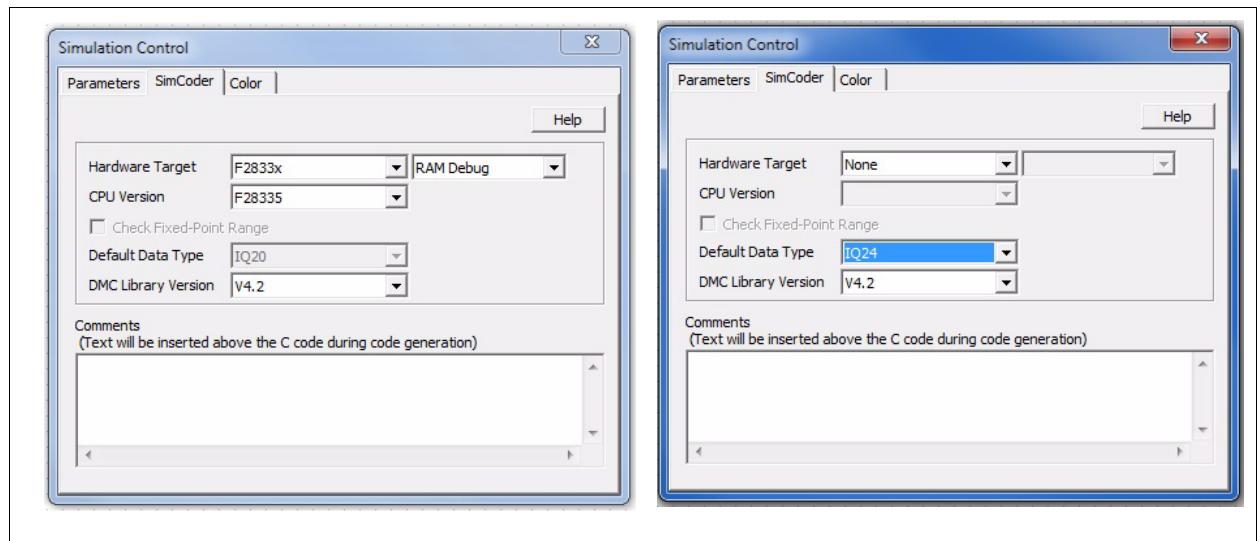


2.5.2 Subcircuit Code Generation for Hardware Target

The same example circuit in Section 2.5.1 can also be used to generate code for targeted hardware.

One sets the desired hardware target type in the Simulation Control dialog's SimCoder tab. As shown below on the left, the target is selected for F2833x, with CPU version of F28335 for RAM Debug build.

One can select *None* as the hardware target but can still generate targeted code for a specific default data type. The example below on the right set the data type as Fixed Point with the position at IQ24.



Go to the subcircuit's attributes dialog. Click on the **Generate Code** button and select **Generate Code for Hardware Target**.

Below is the C code generated by the SimCoder for the subcircuit for F28335 CPU. Unlike the generated code for the whole system, the code for a subcircuit does not have the main program and the initialization routine. It can be inserted into one's own code for F28335 target hardware implementation.

This subcircuit has only one sampling rate. As a result, the generated code has only one function *TaskS1*. The variables *fIn0* refers to the subcircuit input *I_fdbk*, and the variable *fOut0* corresponds to the subcircuit output *Ctrl*.

```

#defineNULL (0)
#ifndef DSP28_DATA_TYPES
#define DSP28_DATA_TYPES
typedef int int16;
typedef long int32;
typedef long long int64;
typedef unsigned int Uint16;
typedef unsigned long Uint32;
typedef unsigned long long Uint64;
typedef float float32;
typedef long double float64;
#endif
DefaultTypefGblS1_U1 = 0;
DefaultTypefGblS1_iref = 0;

// Parameter list for S1
void TaskS1(DefaultType fIn0, DefaultType *fOut0)
{
    DefaultTypefS1_U1, fS1_SUMP1, fS1_B4, fS1_k2, fS1_k1, fS1_SUM1, fS1_Z1,
    DefaultTypefS1_VDC2;

    *fOut0 = fGblS1_U1;

    fS1_VDC2 = 2;
    fS1_Z1 = fIn0;
    fS1_SUM1 = fS1_VDC2 - fS1_Z1;
    fS1_k1 = fS1_SUM1 * 0.4;
    fS1_k2 = fS1_SUM1 * 1000;
    {
        static DefaultType out_A = 0;
        fS1_B4 = out_A + 1.0/20000 * (fS1_k2);
        out_A = fS1_B4;
    }
    fS1_SUMP1 = fS1_k1 + fS1_B4;
    fGblS1_U1 = fS1_SUMP1;
#ifdef_DEBUG
    fGblS1_iref = fS1_VDC2;
#endif
}

```

2.5.3 Restrictions for Subcircuit Code Generation

There are a few restrictions for building the subcircuit for code generation using SimCoder. These restrictions are listed below:

- All the elements in the sub-system must be supported for code generation. To find out if an element can be used for code generation, in PSIM, go to **Options >> Settings**, and check the box **Show image next to elements that can be used for code generation**. Any elements that have an image next to the elements in the PSIM Elements library can be used for code generation.
- Only uni-directional subcircuit ports can be used. That is, input signal ports must be used for subcircuit inputs, and output signal ports must be used for subcircuit outputs. Bi-directional ports are not allowed.
- No hardware input/output elements (such as A/D converter, digital input/output, encoder, counter, and PWM generator) as well as hardware interrupt elements are placed inside the subcircuit. They must be in the top-level main circuit only.
- If the input of a sub-system has a sampling rate, and the rate can not be derived from the circuit inside the sub-system, a zero-order-hold block must be connected at the input to explicitly define its sampling rate. If the zero-order-hold block is not used in this case, this input (and subsequent blocks that connect to this input) will be treated with no sampling rate.
- If the input of a subcircuit does not have a zero-order-hold block connected to it, SimCoder will derive its sampling rate from the blocks that connect to it in the sub-system. However, to avoid ambiguity, it is strongly suggested to place a zero-order-hold block at each input to explicitly define its sampling rate.

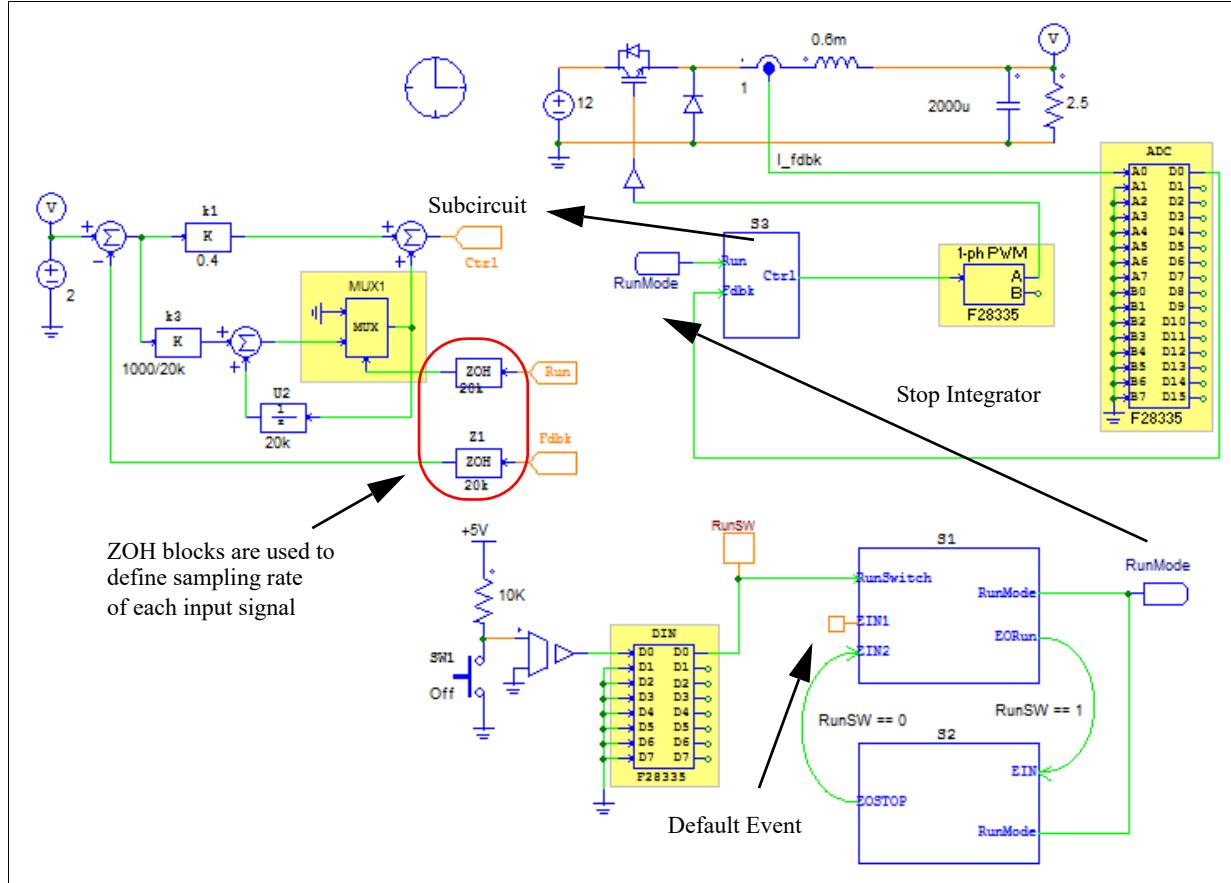
2.6 System with Event Control

Often a system may include event transitions. The system will transit from one state to another state when certain condition is met. SimCoder handles the event control through subcircuits. More detailed description of the event control can be found in Chapter 3.

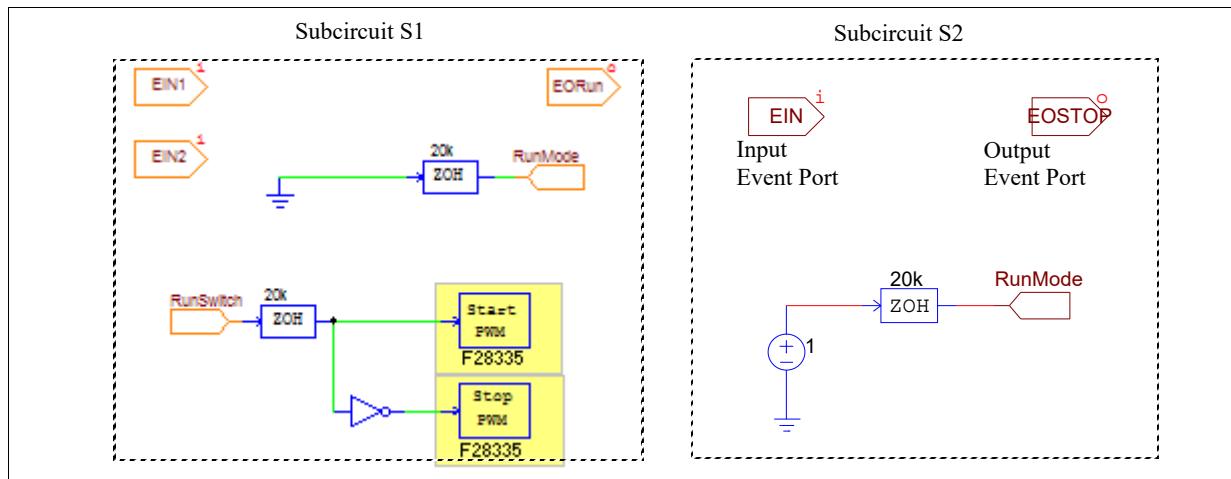
To illustrate how event control works, the following considerations are added to the example in Section :

- A manual switch is employed to control the start/stop of the system. As a result, the system will have two operation modes: Stop Mode and Run Mode. The system will transit from one mode to another whenever the switch position changes.
- In Stop Mode, the integrator output is reset to 0 to prevent saturation.

The example system with event control is shown below.



In the diagram, Blocks S1 and S2 are subcircuits, and the contents of the subcircuits are shown below.



Comparing with the circuit Section , the following changes are made:

- Two event control subcircuits are added to implement the two operation modes: Stop Mode (represented by Subcircuit S1) and Run Mode (represented by Subcircuit S2).
- The default mode of operation is the Stop Mode. This is defined by connecting the **Default Event** element to Port *EIN1* of the subcircuit S1.
- Subcircuit S1 has two input event ports *EIN1* and *EIN2*, one output event port *EORun*, one input signal port *RunSwitch*, and one output signal port *RunMode*. Subcircuit S2 has one input event port *EIN*, one output event port *EOSTOP*, and one signal port *RunMode*.
- Conditions are defined for the transition from the Stop Mode to the Run Mode, and vice versa. The variable *RunSW* used in the conditions is a global variable (refer to Section 5.2 for more details), and is defined by the global variable element connected to the output pin D8 of the digital input element.
- The hardware digital input element is used to measure the position of the push-button switch SW1. When the switch is off, the digital input voltage is high (1), so is the global variable *RunSW*, and the system is in the Run Mode. When *RunSW* is low (0), the system is in the Stop Mode.
- Multiplexer MUX1 is added to prevent the integrator from integrating in the Stop Mode. When the system is not running, the signal *RunMode* will be 0 and the integrator will not integrate. When the signal *RunMode* is 1, the integrator will start to work.

Below is how this system works:

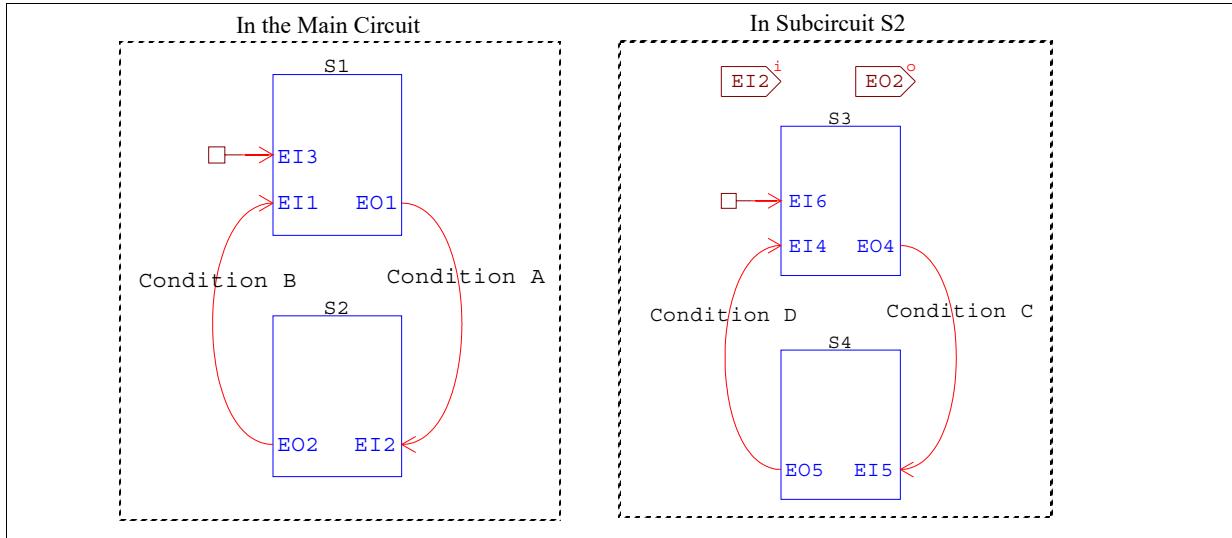
- The position of the manual switch is read through the hardware digital input. This signal is sent to Subcircuit S1 (Stop Mode) through the input signal port *RunSwitch*. The same signal is also designated as the global variable *RunSw*.
- Initially the system is in the Stop Mode. When the condition "*RunSW == 1*" (or *RunSW* is equal to 1) is met, the system will transit from the Stop Mode to the Run Mode. This is defined by the connection of the output event port *EORun* of S1 to the input event port *EIN* of S2.
- While in the Run Mode, if the condition "*RunSW == 0*" (or if *RunSw* is equal to 0) is met, the system will transit from the Run Mode to the Stop Mode. This is defined by the connection of the output event port *EOSTOP* of S2 to the input event port *EIN2* of S1.
- In the Stop Mode subcircuit, the *RunMode* signal will be set to 0. As long as the *RunSwitch* signal is 0, the hardware PWM generator will be stopped. But when the *RunSwitch* is changed to 1, it will start PWM, and at the same time switch out of the Stop Mode into the Run Mode.
- In the Run Mode subcircuit, the *RunMode* signal will be set to 1 in order to enable the integrator operation.

After the system is modified, user may run the simulation to verify that the changes are correct before generating system code for target hardware operation.

Event Handling

3.1 Basic Concept

Event is used to describe the transition of a system from one operation state to another. For example, the figure below shows several operation states and how the transition occurs.



In the main circuit, there are two states: S1 and S2, both in the form of subcircuits. The schematic of each state is included in a subcircuit. State S1 has two input event ports: $EI1$ and $EI3$, and one output event port $EO1$. State S2 has one input event port $EI2$ and one output event port $EO2$. By default, State S1 is the default state at the beginning. This is defined by the connection of the default event element to the input event port $EI3$.

The output event port $EO1$ of S1 is connected to the input event port $EI1$ of S2, with the transition Condition A. This means that when Condition A is met, the system will transit from State S1 to S2. Similarly, the output event port $EO2$ of S2 is connected to the input event port $EI1$. When Condition B is met, the system will transit from State S2 to S1.

When two or more states can not co-exist and only one state can exist at any time, such as S1 and S2 in this case, we refer to these states as exclusive states.

The system on the right shows the content inside Subcircuit S2. It in turn has two states, S3 and S4. When the system transits to State S2, it will start with State S3 by default. If Condition C is met, it will transit from State S3 to S4. If Condition D is met, it will go back to State S3.

There is no limit on the number of states that a system can contain.

3.2 Elements for Event Handling

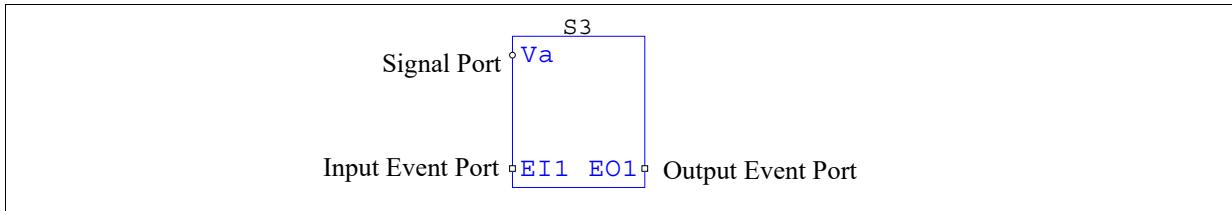
The following elements are used to define events and the state transition:

- Input event port
- Output event port
- Default event element
- Flag for event block first entry
- Hardware interrupt element (see Section 5.4)
- Global variable

Placing an input event port inside a subcircuit will create an event that allows the transition into the subcircuit. Similarly, placing an output event port inside a subcircuit will create an event that allows the transition out of

the subcircuit.

For example, the figure below shows the image of a subcircuit after an input event port and an output event port are placed inside the subcircuit.



The image of an event port is a square, which is different from the image of a signal port which is a circle.

The connection to an input event port can only come from an output event port or a hardware interrupt element, using the event connection wiring function. Input/output event ports and hardware interrupt elements can not be connected to other types of nodes.

For each output event port, a condition must be defined. The property window of the output event port *EO1* in Subcircuit *S3* above, for example, is shown below:



The condition "RunFlag == 1" is the condition that will trigger the output event to occur. The condition statement must be a valid C code expression. For example, the condition statement can be:

`(RunFlag == 1) && (FlagA >= 250.) || (FlagB < Vconst)`

Note that only global variables, numerical values, and parameter constants defined in parameter files or passed from the main circuit into subcircuits can be used in the condition expression. In the above expression, *RunFlag*, *FlagA*, and *FlagB* can be global variables, and *Vconst* can be a constant defined in a parameter file or passed into the subcircuit from the main circuit.

To create a global variable, connect the global variable element to a node.

3.3 Restrictions on Subcircuits with Events

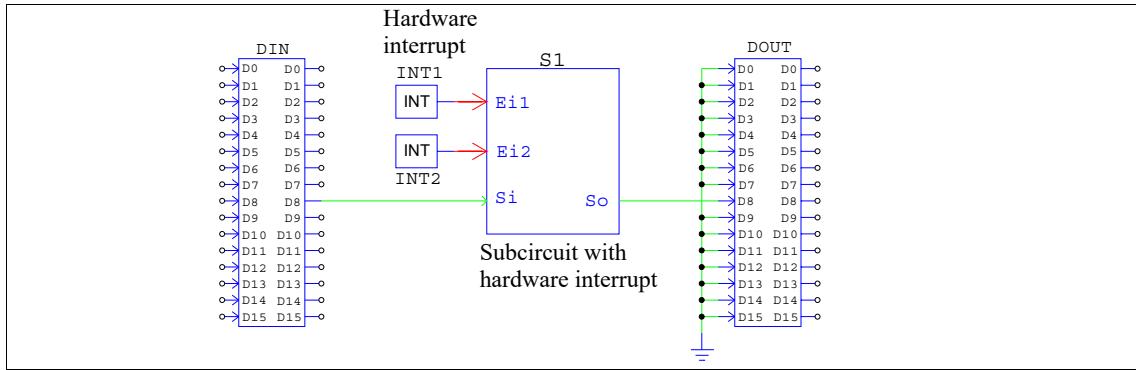
A subcircuit that contains input or output event ports is considered to be a subcircuit with events. Also, everything inside a subcircuit with events will inherit the event property. That is, if Subcircuit A is within Subcircuit B, and Subcircuit B is a subcircuit with events, even if Subcircuit A does not have any input/output event ports, it is still considered as a subcircuit with events.

As subcircuits are used to handle events, there are now three types of subcircuits in PSIM:

- *Regular subcircuits*: This type of subcircuit does not contain any event ports and is the same as conventional subcircuits before.
- *Subcircuit with events*: This type of subcircuit contains input/output events ports, but there are no hardware interrupt elements connected to the input event ports.
- *Subcircuit with hardware interrupt*: This type of subcircuit contains input event ports only, and only hardware interrupt elements are connected to the input event ports. There is no output event port inside the subcircuit, and no output event ports are connected to the input event ports. This is a special case of the subcircuit with events as this type of subcircuit is dedicated to handle hardware interrupt only.

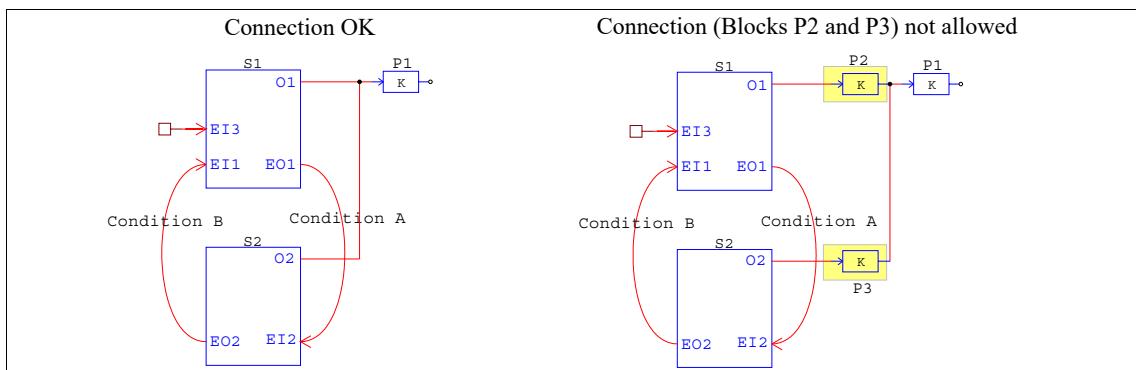
Since a subcircuit with events or with hardware interrupt is involved in the code generation only, the following restrictions are imposed on these two types of subcircuits:

- All the elements inside a subcircuit with events or with hardware interrupt must be supported by code generation. For example, such a subcircuit can not contain a resistor or a rms block, both not supported by the code generation.
- A subcircuit with hardware interrupt can have multiple input event ports, but can not have any output event ports. Also, only hardware interrupt elements can be connected to the input event ports. In addition, the signal input/output ports of the subcircuit can be connected to hardware elements only, not to other function blocks. The figure below shows how a subcircuit with hardware interrupt can be connected:



The subcircuit S1 is a subcircuit with hardware interrupt. It has two hardware interrupt elements connected to it, INT1 and INT2. It has one signal input port S_i connected to the hardware digital input, and one signal output port S_o connected to the hardware digital output.

- If a subcircuit with hardware interrupt contains z-domain blocks with sampling rates, these sampling rates will be ignored as the subcircuit will be called only when a hardware interrupt occurs. For example, if the subcircuit contains a discrete integrator, the sampling rate of the discrete integrator will be ignored. In the calculation for the integrator, the previous time will be the last time that a hardware device triggers an interrupt.
- If the signal outputs of two subcircuits are connected, they should be connected directly, not through other elements. To illustrate this, consider the following circuits:



In the circuit on the left, both subcircuits S1 and S2 have one output signal port, O1 and O2. They are connected externally together to the input of the proportional block P1. The way the circuit works is that the input of the block P1 will come from either Port O1 or O2, depending on which state is active. This connection is allowed. However, in the circuit on the right, Port O1 is connected to Block P2, and Port O2 is connected to P3, and the outputs of P2 and P3 are then connected together. Such a connection is not allowed. In this case, Block P2 should be moved into the subcircuit S1, and Block P2 moved into the subcircuit S2.

SimCoder Libraries

4.1 Overview

SimCoder can be used with or without a hardware target. When it is used without a hardware target, it will convert a control schematic into C code. While the code can be simulated in PSIM, it is not for a specific hardware. On the other hand, with a hardware target, SimCoder can generate code that is ready to run on the specific hardware, or can be adopted for a specific hardware.

SimCoder element libraries include two types of elements: these that are not associated with any hardware targets or are shared by all hardware targets, and these that are specific to a particular hardware.

SimCoder elements that are independent of any hardware include the following:

- Some of the elements of the standard PSIM library.
- All the elements under **Elements >> Event Control**.
- The *Global Variable* element under **Elements >> SimCoder**.

Simcoder elements that are shared by all hardware targets include the following:

- The *Interrupt* element under **Elements >> SimCoder**.
- The *TI DMC* element under **Elements >> SimCoder >> TI DMC Library**.

SimCoder elements that are hardware-specific include the following:

- F2833x Target: All the elements under **Elements >> SimCoder >> F2833x Target**.
- F2803x Target: All the elements under **Elements >> SimCoder >> F2803x Target**.
- F2806x Target: All the elements under **Elements >> SimCoder >> F2806x Target**.
- F2802x Target: All the elements under **Elements >> SimCoder >> F2802x Target**.
- PE-Expert4 Target: All the elements under **Elements >> SimCoder >> PE-Expert4 Target**.

The SimCoder elements that are independent or common to all hardware targets are described in this Chapter. The elements for each specific hardware target are described in Chapter 6 to 9. Elements in the TI DMC Library are described in Chapter 13.

4.2 Elements from Standard PSIM Library

Many elements in the standard PSIM library can be used for code generation. Under **Options >> Settings >> Advanced**, check the option box "Show image next to elements that can be used for code generation", a small image  will appear next to those elements that can be used for code generation.

Please note that, for all the math function blocks and the Simplified C Block, variables t (for time) and delta (for time step) can not be used in SimCoder for code generation.

Also, the parameter file element and the sawtooth voltage source element have special usage in SimCoder, as described in the sections below.

4.2.1 Defining Global Parameters in Parameter File

The parameter file element can be used in the same way as before. In SimCoder, it can also be used to define global parameters.

In order to make generated code more readable and manageable, sometimes it is better to use parameter names instead of the actual numerical values in the code. For example, if the gain of a controller is 1.23, rather than using the number 1.23 in the code, we can define the parameter $K_p = 1.23$, and use the parameter name K_p in the code instead.

This type of parameters is global to the code, and can be used anywhere in the code. To define a parameter as a global parameter, in the content of a parameter file, use the "(global)" definition before the parameter name.

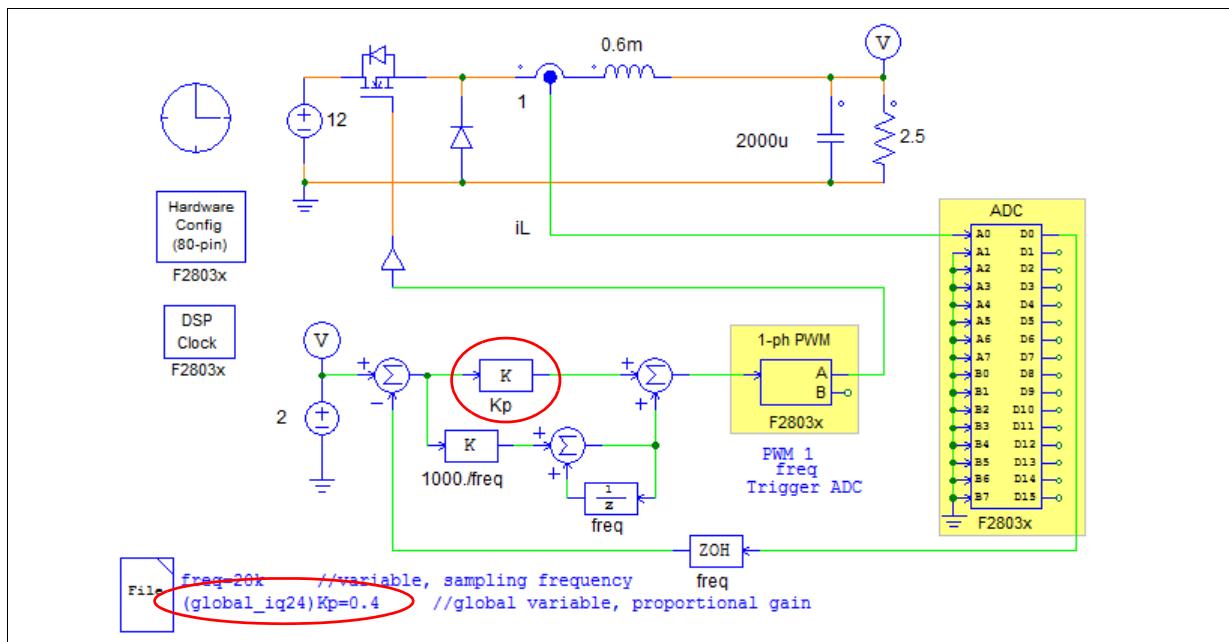
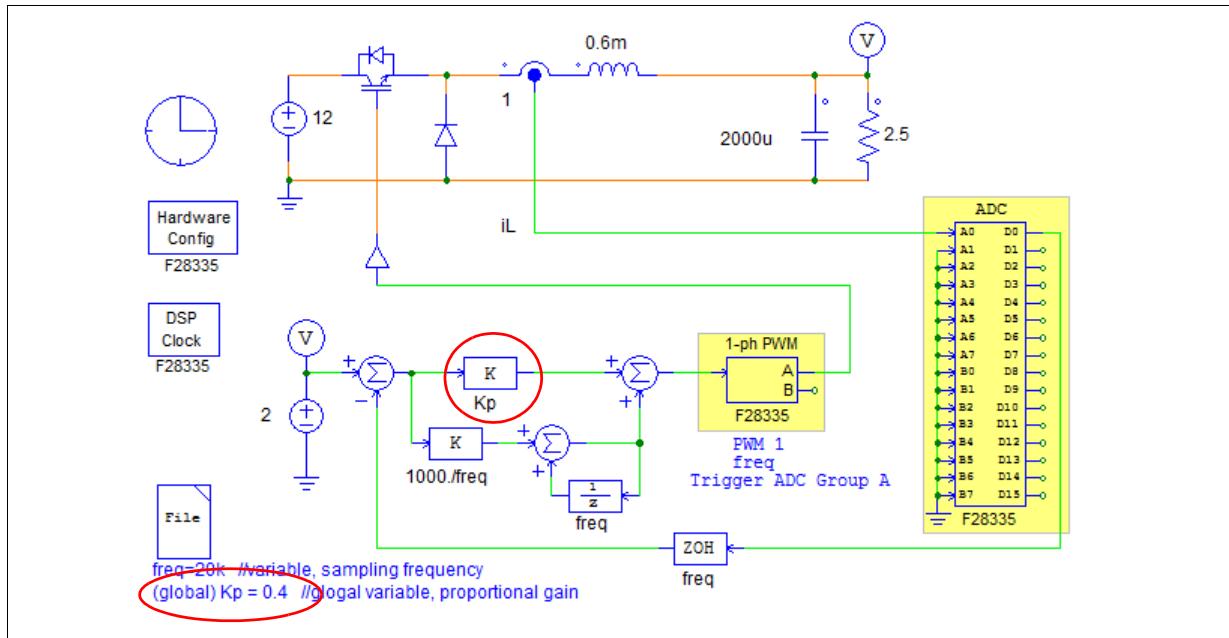
As shown below in the buck converter example circuit, the gain of the proportional controller is defined as K_p .

In the parameter file, if the DSP target is floating-point F28335, the parameter Kp is defined as:

$$(\text{global}) \text{Kp} = 0.4$$

If the DSP target is F28035 with fixed-point at IQ-24, the parameter Kp is defined as:

$$(\text{global_Iq24}) = 0.4$$



The generated code for F28335 is shown below. Note that in the code, the parameter Kp is defined as 0.4 at the beginning, and the parameter name Kp is used in the calculation.

```

/
*****
// This code is created by SimCoder Version 9.1 for TI F28335 Hardware Target
//
// SimCoder is copyright by Powersim Inc., 2009-2011
//
// Date: November 08, 2011 11:26:37
*****
*****/
#include<math.h>
#include"PS_bios.h"
typedef float DefaultType; The global parameter Kp is defined here.
#define GetCurTime() PS_GetSysTimer()

interrupt void Task();

DefaultTypefGbliref = 0.0;
DefaultTypefGblUDELAY1 = 0;

DefaultTypeKp = 0.4;
interrupt void Task()
{
    DefaultType fVDC2, fTI_ADC1, fZOH3, fSUM1, fP2, fSUMP3, fUDELAY1,
    fP1, fSUMP1; The parameter Kp is used here.
    PS_EnableIntr();
    fUDELAY1 = fGblUDELAY1;
}

```

4.2.2 Generating Sawtooth Waveform

A sawtooth waveform can be used in hardware as the system timer, or to generate other periodic waveforms (such as sinusoidal waveform). The sawtooth voltage source under **Elements >> Sources >> Voltage** is implemented using an actual counter in the hardware.

It assumes that there exists a 32-bit counter in the hardware, incrementing in every 20 ns, to generate the sawtooth waveform.

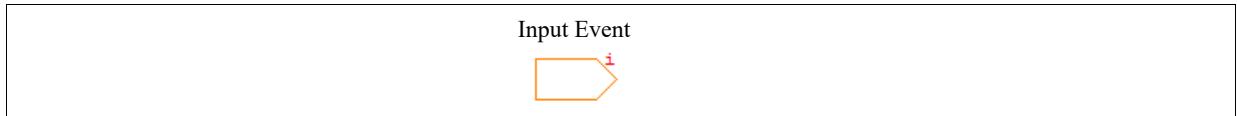
For the PE-Expert4 Hardware Target, it used the 32-bit free-run counter on the PEV Board, incrementing in every 20 ns, to generate the sawtooth waveform.

4.3 Event Control Elements

The following elements are used to implement event control.

4.3.1 Input Event

The image of an input event element is shown below.

Image:

The letter "i" in the image refers to "input".

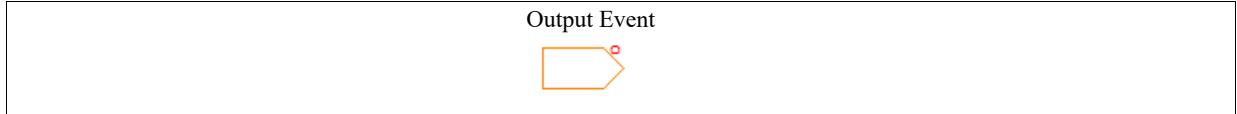
The input event element is a type of subcircuit interface port. It should be used inside a subcircuit only. After double clicking on the element, one will define the port name and location, as shown below:



In the main circuit that calls this subcircuit, if there is an event connection wire connecting to this port, when the condition of the event connection is met, the system will transit to this subcircuit through this input event port.

4.3.2 Output Event

The image of an output event element is shown below.

Image:

The letter "o" in the image refers to "output".

The output event element is also a type of subcircuit interface port. It should be used inside a subcircuit only. After double clicking on the element, one will define the port name, location, as well as a condition, as shown below:



When the condition defined in the output event port is met, the system will transit out of this subcircuit into another subcircuit.

The condition statement must use format and operators supported by the C language. For example, the statements below are acceptable condition statements:

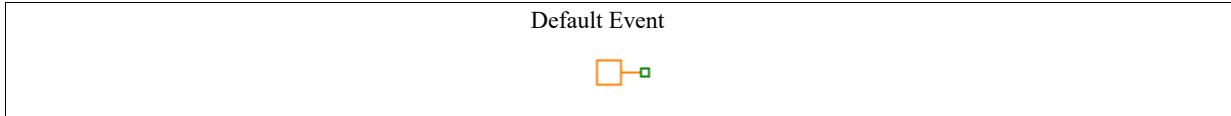
```
A == 1  
A >= B  
(A > B) && (C > D)
```

where A, B, C, and D are global variables or numerical constants.

4.3.3 Default Event

The image of a default event element is shown below.

Image:



When there are several exclusive states, the default event element is used to define which state is the default state. It is connected to the input event port of a subcircuit outside the subcircuit.

4.3.4 Event Connection

The event connection is a SimCoder element which connects an output event port or a hardware interrupt element to an input event port. Note that it should not be confused with the regular wiring tool to connect other PSIM elements. The event connection can be used for event connection only.

Event connection element can be found at **Elements >> Event Control >> Event Connection**.

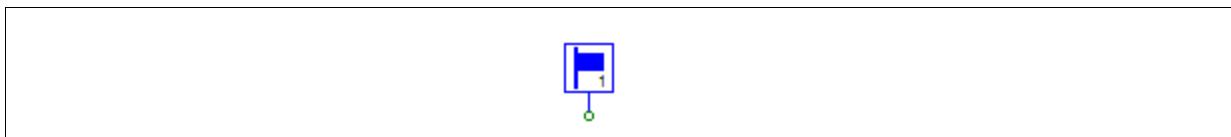
When double clicking on the event connection wire, one can edit the condition statement of the output event port that the event wire connects to.

Besides the starting point and the ending point, an event connection wire has two points in between. By modifying the locations of these two points, the shape of the connection wire can be changed. To modify these two points, highlight the event connection wire. Right click and choose "Modify handle 1" or "Modify handle 2".

4.3.5 Flag for Event Block First Entry

Sometimes certain actions need to be performed when the program execution enters an event subcircuit block for the first time. To identify this, a flag for event block first entry is provided.

Image:



Attribute:

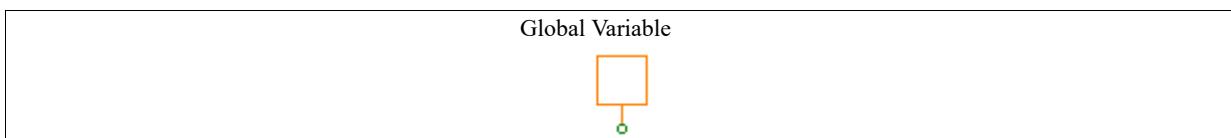
Parameters	Description
Event Subcircuit Block Name	The name of the event subcircuit block that the flag is for.

The flag node is an output node. When the event subcircuit block is entered for the first time, the node value will be 1. Otherwise, it will be 0. For example, to find out when the event subcircuit block S1 is entered the first time, set *Event Subcircuit Block Name* to S1.

4.4 Global Variable

A global variable is used in conditional statements and in special occasions.

Image:



Attributes:

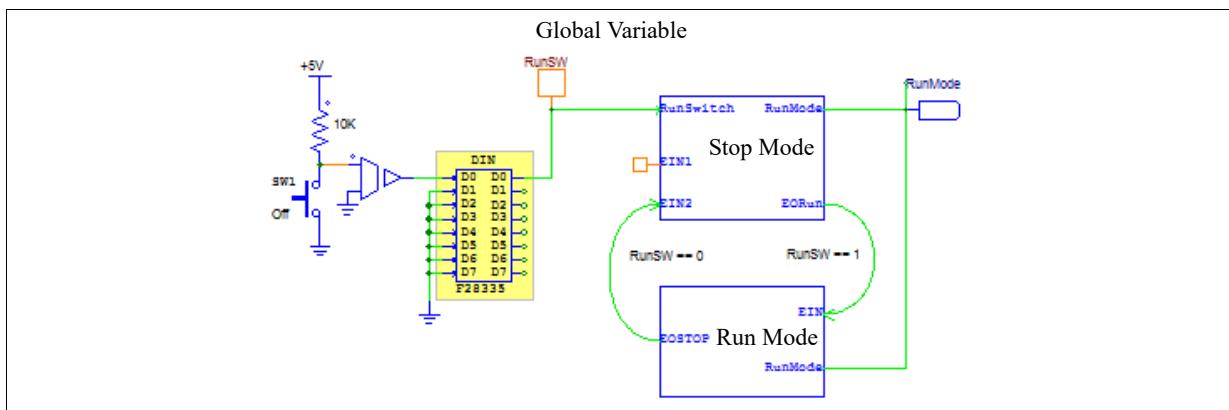
Parameters	Description
Name	The name of the global variable name
Initial Value	The initial value of the global variable

To define a signal as a global variable, connect the global variable element to the particular node. Note that only a signal in the control circuit for the code generation can be defined as a global variable.

As the name suggests, a global variable can be accessed globally. When the initial value of a global variable is changed, the initial values of all the global variables in that circuit, including subcircuits, are changed at the same time.

A global variable can be a signal sink or a signal source. When it is a signal sink, it reads the signal value from the node. When it is a signal source, it sets the value of the node.

One use of the global variables is in the event condition statements. All variables in the condition statements must be global variables. An example is shown below.



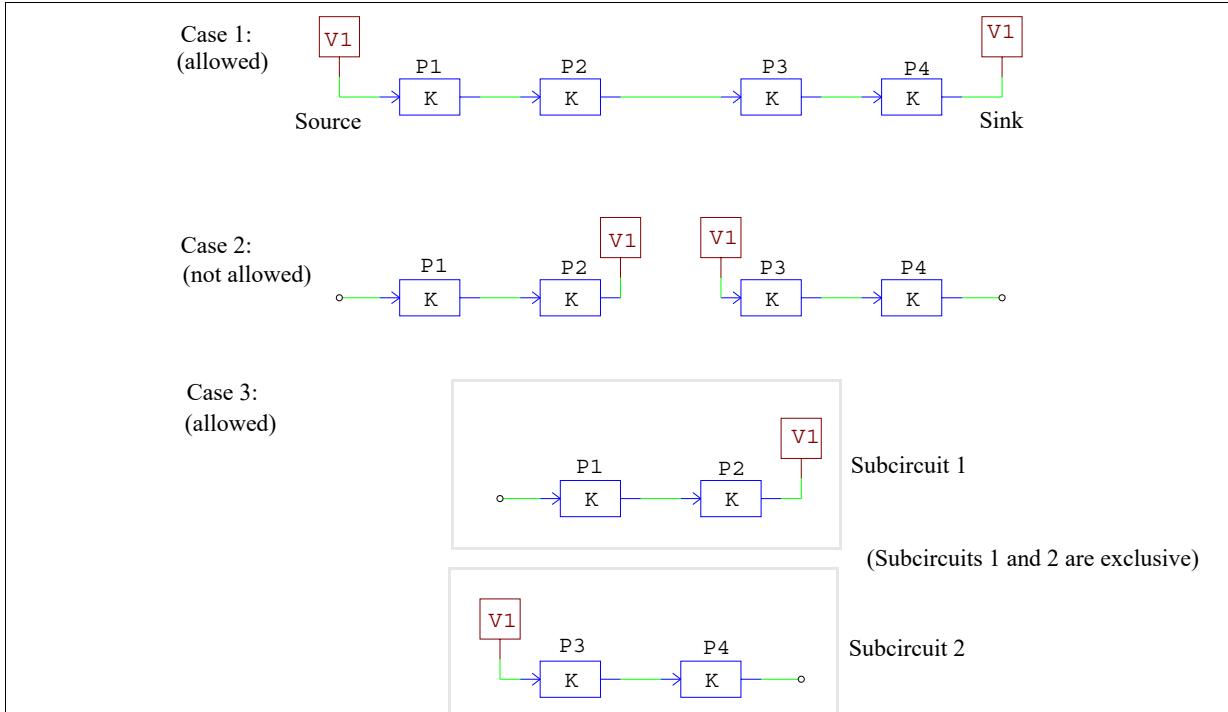
In this example, a global variable, *RunSW*, is connected to the output pin D0 of the digital input. This global variable is then used in the conditional statements between the transition of the two modes of operation.

Another use of the global variable is to use it as a signal source. For example, a global variable can be used as a signal source and passes the value to another block.

Note that global variable should not be used as a label to pass a value from one node to another, when two nodes can be physically connected by a wire. The use of the global variables has the following restrictions:

- Global variables of the same name can be used multiple times only if they are in the same signal flow path.
- If they are in different signal flow paths, global variables of the same name are not allowed, unless they are in different exclusive states (exclusive states are states that can not occur at the same time).

To illustrate this, the diagram below shows situations where global variables can and cannot be used.



In Case 1, a global variable **V1** is first used as a source and it assigns the value to the input of the block **P1**. After a series of calculation, the output of the block **P4** is assigned back to the same global variable **V1**. Since both global variables are in the same signal flow path, it is allowed.

In Case 2, however, the global variable **V1** is used as a label to pass values from the output of the block **P2** to the input of the block **P3**. This is not allowed. To pass the value from one node to another, labels should be used instead, or one should connect these two nodes with a wire.

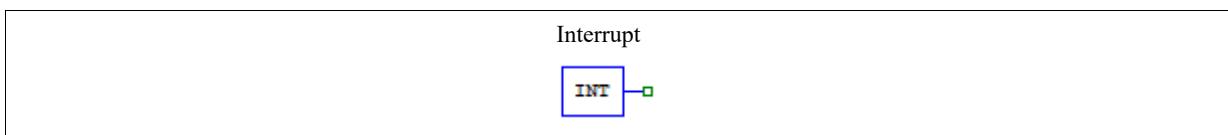
In Case 3, on the other hand, the global variable **V1** is used in both Subcircuits 1 and 2. Subcircuit 1 and 2, however, are two exclusive states. That is, the system will run either Subcircuit 1, or Subcircuit 2, but not both. The use of the global variables is allowed in this case.

4.5 Interrupt

In a hardware target, elements such as digital input, encoder, capture, and PWM generators (for F2833x and F2803x DSPs) can generate hardware interrupt. The interrupt block allows users to associate the element that generates the interrupt with the corresponding subcircuit that represents the interrupt service routine.

Please note that the interrupt element cannot be placed inside a subcircuit. It must be in the top-level main circuit only.

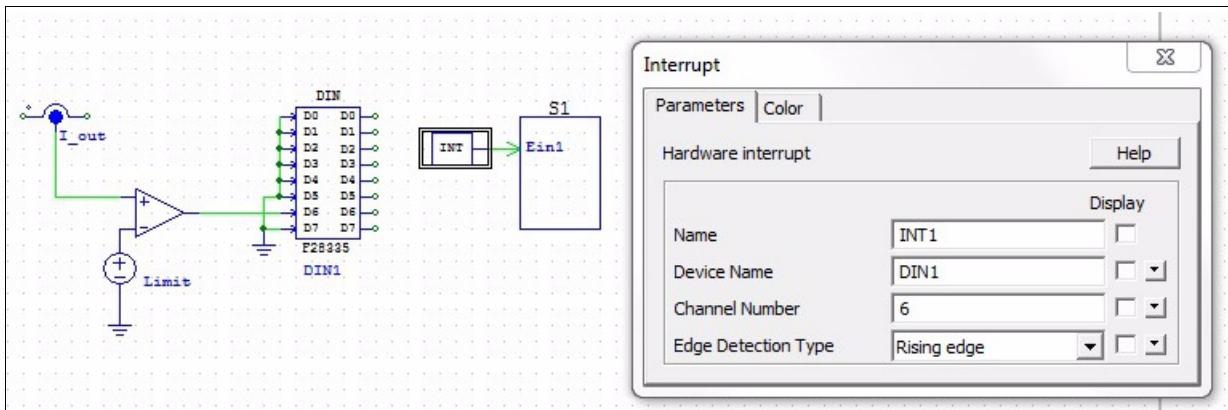
Image:



Attributes:

Parameters	Description
Device Name	The name of the hardware device that initiates the hardware interrupt
Channel Number	The input channel number of the device that initiates the interrupt. For example, if Channel D0 of a digital input generates the interrupt, the channel number should be set to 0. Note that this parameter is used only for: - Digital input - Capture (PE-Expert4 Target only) It does not apply to encoder and PWM generator.
Trigger Type	This applies to digital input and capture only. It can be one of the following: - <i>No edge detection</i> : No interrupt will be generated. - <i>Rising edge</i> : The rising edge of the input signal will generate interrupt. - <i>Falling edge</i> : The falling edge of the input signal will generate interrupt. - <i>Rising/falling edges</i> : Both the rising and falling edges of the input signal will generate interrupt.

The diagram below shows how the interrupt block is used.



In this circuit, the current I_{out} is measured and compared with the reference value $Limit$. If the current I_{out} exceeds the $Limit$, the output of the comparator will change from 0 to 1. This will generate a rising edge to channel $D6$ of the digital input block **DIN1**. The interrupt block parameters are set as shown in the graph:

- Device Name: **DIN1** for the specified digital input block;
- Channel Number: 6 for the specified digital input channel $D6$;
- Edge Detection Type: Rising edge for the condition of $I_{out} > Limit$.

The rising edge at the output of the comparator will then generate a hardware interrupt and the operation will transit to Subcircuit **S1** through the input event port **Ein1**.

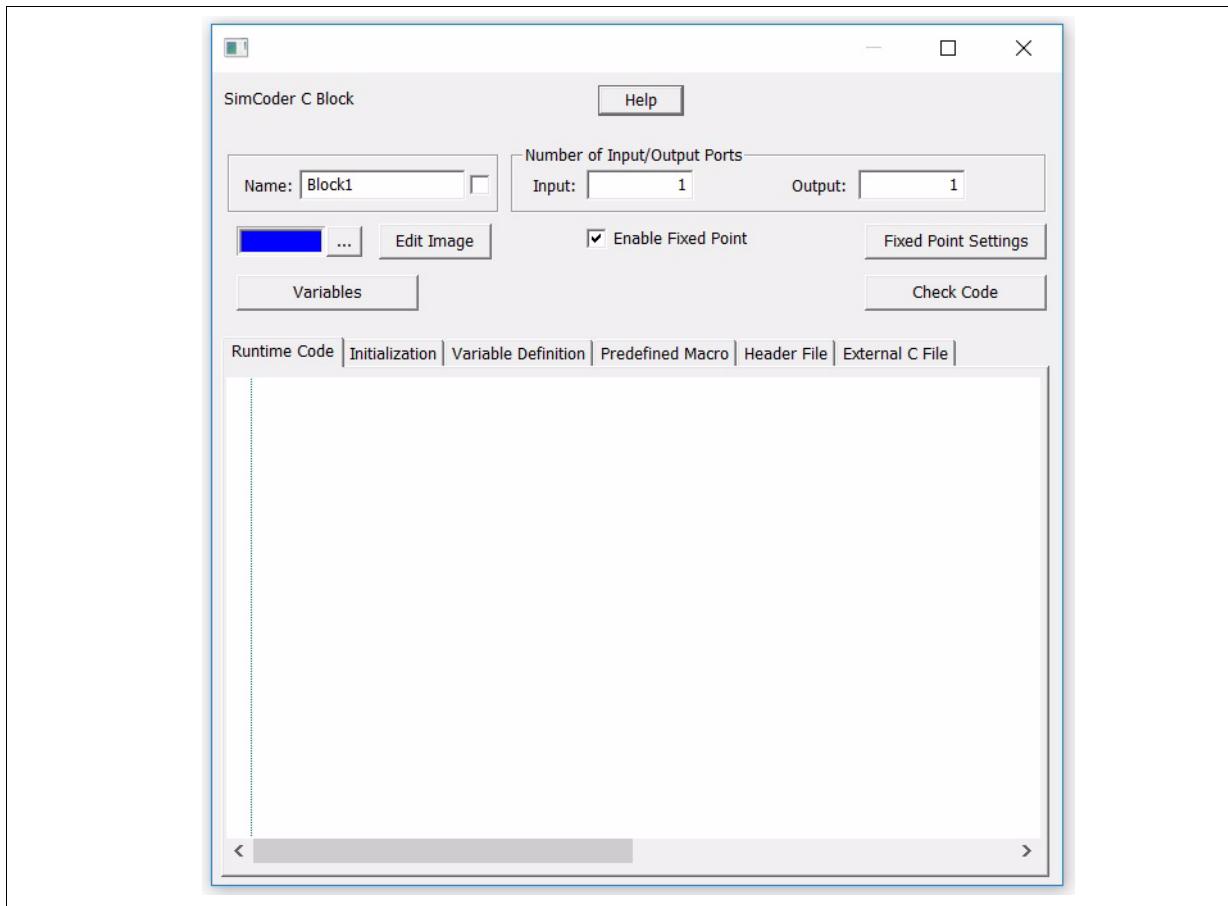
Please note that the connection between the interrupt block **INT1** and the event subcircuit **S1** is an **Event Connection**, not a piece of wire.

4.6 SimCoder C Block

A SimCoder C block is a C block that can accept external header files and C code files, and can combine simulation code and hardware target code easily.

The SimCoder C block is similar to the simplified C block in that it can be used in a SimCoder circuit for hardware code generation. However, unlike the simplified C block, the SimCoder C block provides the functions to easily include external header and C files, perform initialization, and define and call predefined macros. In addition, a flag called "Flag_Simulation" is provided so that users can have both simulation code and hardware code in the C block, and use the flag to differentiate the two versions of the code.

An image of the SimCoder C block dialog window is shown below.



The tabs of the dialog window are defined below.

Tab	Description
Runtime Code	Main code that runs each time
Initialization	The code that runs only once at the beginning for initialization
Variable Definition	Definition of global variables
Predefined Macro	Macros that are defined before external header files or C files to pre-process the code. For example, the IQ math library uses a macro called "MATH_TYPE" to determine if the library is floating-point (MATH_TYPE=1) or fixed-point (MATH_TYPE=0). The macro "#define MATH_TYPE 1" can be defined in this tab.
Header File	External header files .h
External C File	External C files that contain code for function implementation

Other functions in the dialog window include the following:

Function	Description
Edit Image	Edit the image of the C block
Enable Fixed Point	When this box is checked, this block becomes a fixed-point block. Otherwise it is a floating-point block.

Fixed Point Settings	Define the data format of the C block inputs and outputs when it is a fixed-point block.
Variables Macro	<p>Define variables that can be passed from the main circuit into the C code. Variable values could be numbers or formulas. These formulas could contain variables that were defined in parameter files or passed to the schematic from command-line. Formulas are evaluated before simulation starts and they can not contain 't' (time) or input/output values.</p> <p>Example:</p> <pre>a = 15.2 b = Freq / sqrt(2)</pre> <p>where 'Freq' is defined in a parameter file.</p> <p>These variables are compatible with SimCoder. Please note that SimCoder generated code will show the evaluated value, not the formula.</p>
Check Code	Check the syntax of the code

SimCoder pre-processors can be used in the SimCoder C block to generate simplified code. In order to differentiate SimCoder pre-processors from the processors in the C language, a double ## sign is used. The following pre-processors are supported:

```
##if condition
...
##elif condition
...
##else
...
##endif
```

Examples of how the SimCoder C block is used can be found in each Target folder under the "examples/SimCoder" folder.

5.1 Overview

TI's IQmath library provides a list of functions that can port floating-point code to fixed-point code. These functions provide high execution speed and high accuracy.

For more information on the IQmath library, please refer to relevant TI documents.

5.2 IQmath Data Type and Range/Resolution

Inputs and outputs of IQmath functions are 32-bit fixed-point numbers. The data type can have the following:

- _iq GLOABLE_Q format
- _iq1 to _iq30 IQ1 to IQ30 format

The data range and resolution are listed in the table below.

Data Type	Range		Resolution/Precision
	Minimum	Maximum	
_iq30	-2	1.999 999 999	0.000 000 001
_iq29	-4	3.999 999 998	0.000 000 002
_iq28	-8	7.999 999 996	0.000 000 004
_iq27	-16	15.999 999 993	0.000 000 007
_iq26	-32	31.999 999 985	0.000 000 015
_iq25	-64	63.999 999 970	0.000 000 030
_iq24	-128	127.999 999 940	0.000 000 060
_iq23	-256	255.999 999 981	0.000 000 119
_iq22	-512	511.999 999 762	0.000 000 238
_iq21	-1024	1023.999 999 523	0.000 000 477
_iq20	-2048	2047.999 999 046	0.000 000 954
_iq19	-4096	4095.999 998 093	0.000 001 907
_iq18	-8192	8191.999 996 185	0.000 003 815
_iq17	-16384	16383.999 992 371	0.000 007 629
_iq16	-32768	32767.999 984 741	0.000 015 259
_iq15	-65536	65535.999 969 482	0.000 030 518
_iq14	-131072	131071.999 938 965	0.000 061 035
_iq13	-262144	262143.999 877 930	0.000 122 070
_iq12	-524288	524287.999 755 859	0.000 244 141
_iq11	-1048576	1048575.999 511 719	0.000 488 281
_iq10	-2097152	2097151.999 023 437	0.000 976 563
_iq9	-4194304	4194303.998 046 875	0.001 953 125
_iq8	-8388608	8388607.996 093 750	0.003 906 250
_iq7	-16777216	16777215.992 187 500	0.007 812 500
_iq6	-33554432	33554431.984 375 000	0.015 625 000
_iq5	-67108864	67108863.968 750 000	0.031 250 000

_iq4	-134217728	134217727.937 500 000	0.062 500 000
_iq3	-268435456	268435455.875 000 000	0.125 000 000
_iq2	-536870912	536870911.750 000 000	0.250 000 000
_iq1	-1073741824	1 073741823.500 000 000	0.500 000 000

F2833x Hardware Target

6.1 Overview

With the F2833x Hardware Target, SimCoder can generate code that is ready to run on any hardware boards based on Texas Instruments' F2833x floating-point DSP.

The F2833x Hardware Target will work with all F2833x packages. The figures in the next two pages show the pin assignments of the F2833x DSP in the low-profile flat-pack (LQFP) package. The main functions implemented in the F2833x Hardware Target are marked in color in the figures.

The F2833x Hardware Target library includes the following function blocks:

- PWM generators: 3-phase, 2-phase, 1-phase, and APWM
- Variable frequency PWM
- Start/Stop functions for PWM generators
- Trip-zone and trip-zone state
- A/D converter
- Digital input and output
- SCI configuration, input, and output
- SPI configuration, device, input, and output
- CAN configuration, input, and output
- Capture and capture state
- Encoder and encoder state
- Up/Down counter
- Interrupt time
- DSP clock
- Hardware configuration

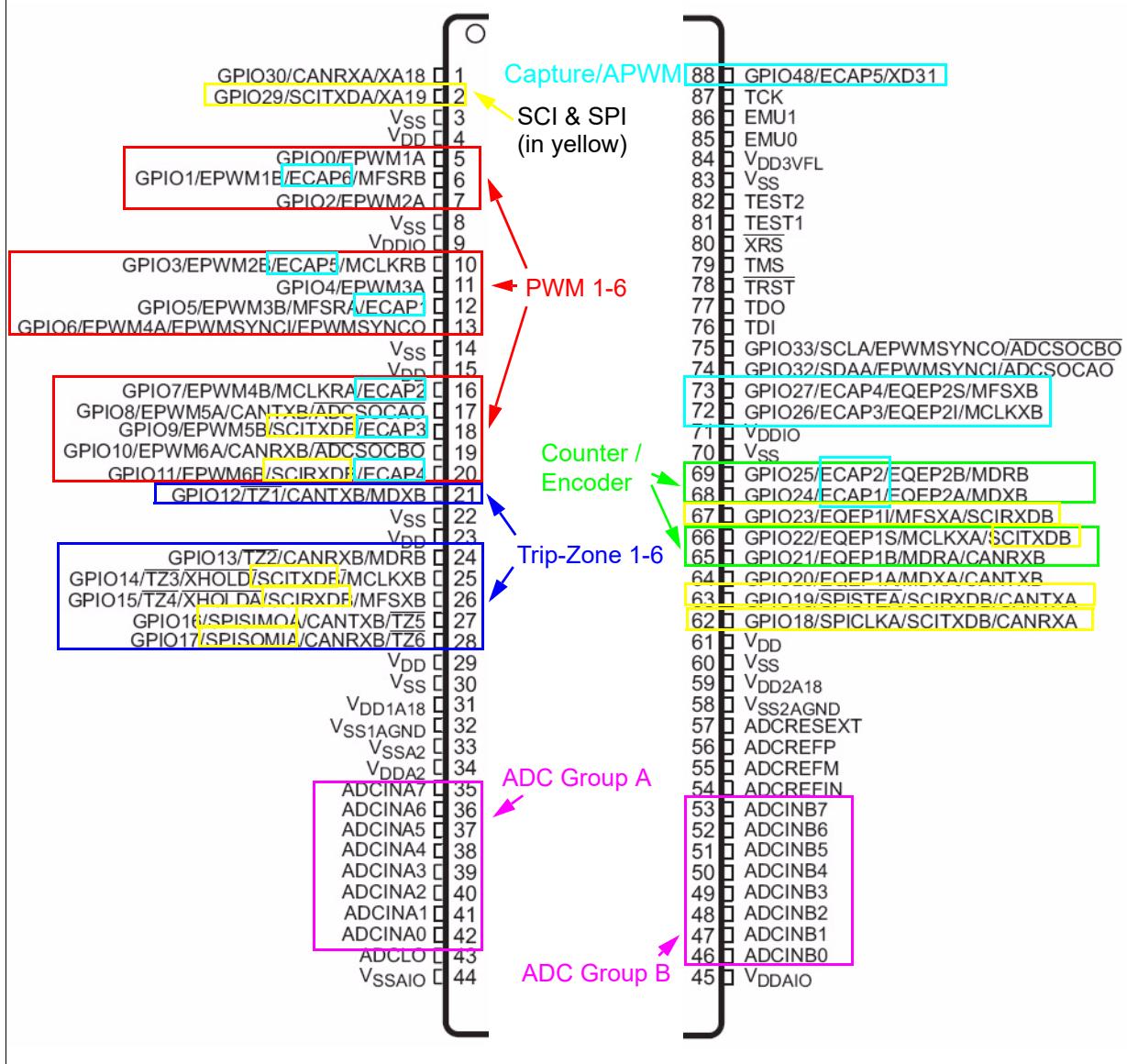
When generating the code for a system that has multiple sampling rates, SimCoder will use the interrupts of the PWM generators for the PWM sampling rates. For other sampling rates in the control system, it will use the Timer 1 interrupt first, and then Timer 2 interrupt if needed. If there are more than three sampling rates in the control system, the corresponding interrupt routines will be handled in the main program by software.

PWM generators can generate hardware interrupt. SimCoder will search and group all the elements that are connected to the PWM generator and have the same sampling rate as the PWM generator. These elements will be automatically placed and implemented in an interrupt service routine in the generated code.

In addition, digital input, encoder, capture, and trip-zone can also generate hardware interrupt. Each hardware interrupt must be associated with an interrupt block (described in Section 5.4 of this Manual), and each interrupt block must be associated with an interrupt service routine (a subcircuit that represents the interrupt service routine). For example, if a PWM generator and a digital input both generate interrupt, there should be one interrupt block and one interrupt service routine for each of them.

The definitions of the elements in the F2833x Hardware Target library are described in this Chapter.

F28335 DSP Port Assignments (Pin 1 - 88)



F28335 DSP Port Assignments (Pin 89 - 176)

132	GPIO75/XD4	133	GPIO76/XD3
131	GPIO74/XD5	134	GPIO77/XD2
130	GPIO73/XD6	135	GPIO78/XD1
129	GPIO72/XD7	136	GPIO79/XD0
128	GPIO71/XD8	137	GPIO38/XWE0
127	GPIO70/XD9	138	XCLKOUT
126	V _{DD}	139	V _{DD}
125	V _{SS}	140	V _{SS}
124	GPIO69/XD10	141	GPIO28/SCIRXDA/XZCS6
123	GPIO68/XD11	142	GPIO34/ECAP1/XRFADY
122	GPIO67/XD12	143	V _{DDIO}
121	V _{DDIO}	144	V _{SS}
120	V _{SS}	145	GPIO36/SCIRXDA/XZCS0
119	GPIO66/XD13	146	V _{DD}
118	V _{SS}	147	V _{SS}
117	V _{DD}	148	GPIO35/SCITXDA/XR/W
116	GPIO65/XD14	149	XRD
115	GPIO64/XD15	150	GPIO37/ECAP2/XZCS7
114	GPIO63/SCITXDC/XD16	151	GPIO40/XA0/XWE1
113	GPIO62/SCIRXDC/XD17	152	GPIO41/XA1
112	GPIO61/MFSRB/XD18	153	GPIO42/XA2
111	GPIO60/MCLKRB/XD19	154	V _{DD}
110	GPIO59/MFSRA/XD20	155	V _{SS}
109	V _{DD}	156	GPIO43/XA3
108	V _{SS}	157	GPIO44/XA4
107	V _{DDIO}	158	GPIO45/XA5
106	V _{SS}	159	V _{DDIO}
105	XCLKIN	160	V _{SS}
104	X1	161	GPIO46/XA6
103	V _{SS}	162	GPIO47/XA7
102	X2	163	GPIO80/XA8
101	V _{pp}	164	GPIO81/XA9
100	GPIO58/MCLKRA/XD21	165	GPIO82/XA10
99	GPIO57/SPISTEA/XD22	166	V _{SS}
98	GPIO56/SPICLKA/XD23	167	V _{DD}
97	GPIO55/SPIISOMIA/XD24	168	GPIO83/XA11
96	GPIO54/SPISIMOA/XD25	169	GPIO84/XA12
95	GPIO53/EQEP1I/XD26	170	V _{DDIO}
94	GPIO52/EQEP1S/XD27	171	V _{SS}
93	V _{DDIO}	172	GPIO85/XA13
92	V _{SS}	173	GPIO86/XA14
91	GPIO51/EQEP1B/XD28	174	GPIO87/XA15
90	GPIO50/FQFP1A/XD29	175	GPIO39/XA16
89	GPIO49/ECAP6/XD30	176	GPIO31/CANTXA/XA17

SCI & SPI (in yellow) → Capture/APWM → Counter/Encoder → Capture/APWM

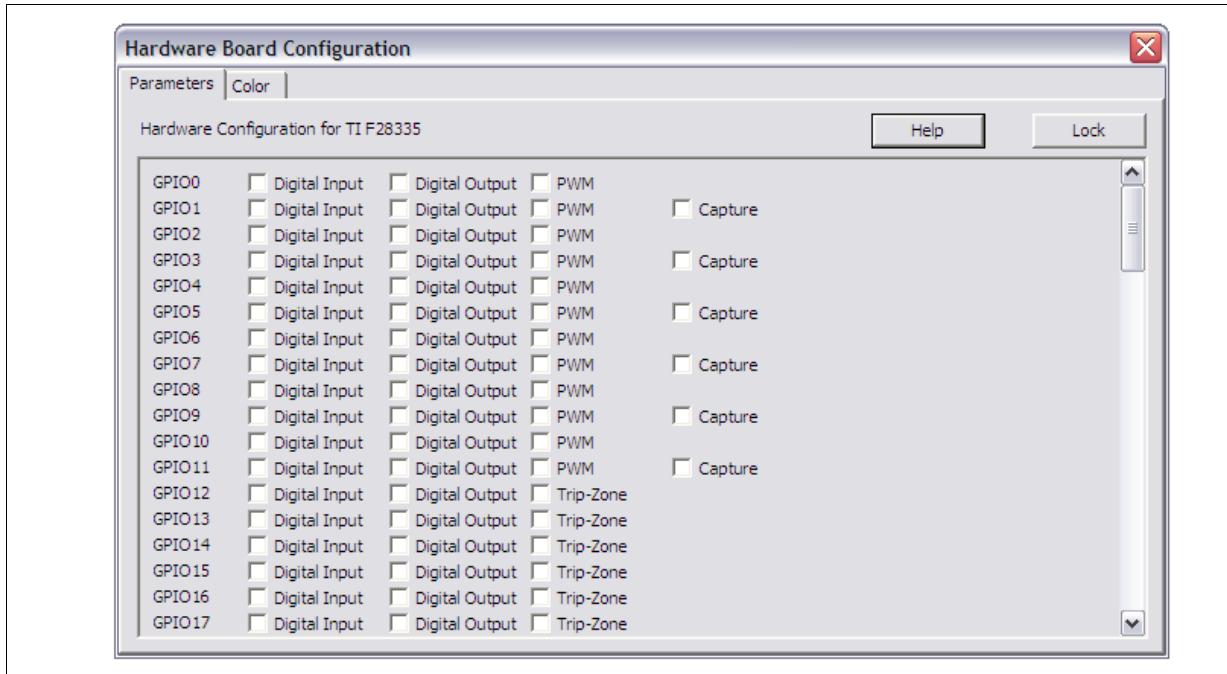
6.2 Hardware Configuration

F2833x provides 88 GPIO ports (GPIO0 to GPIO87), and each port may be configured for different functions. For a particular DSP board, however, not all the ports are accessible from outside, and often the functions of some ports are fixed. The Hardware Configuration block provides a way to configure SimCoder for a particular DSP board.

Image:



The dialog window of the block is shown below:



For each GPIO port, a check box is provided for each of its available function. If this box is checked, only this function is used, and all other functions are not allowed in SimCoder. For example, Port GPIO1 can be used for "Digital Input", "Digital Output", "PWM" and "Capture". If a particular board uses Port GPIO1 as the "PWM" output, only the checkbox for "PWM" should be checked and all other check boxes should be left unchecked. If in the circuit Port GPIO1 is used as "Digital Input", SimCoder will report an error.

6.3 DSP Clock

The DSP Clock block defines the external clock frequency and the speed of the F2833x DSP, as well as the program space size.

Image:



Attributes:

Parameters	Description
External Clock (MHz)	Frequency of the external clock on the DSP board, in MHz. The frequency must be an integer, and the maximum frequency allowed is 30 MHz.
DSP Speed (MHz)	DSP Speed, in MHz. The speed must be an integer, and must be an integer multiple of the external clock frequency, from 1 to 12 times. The maximum DSP speed allowed is 150 MHz.

If the DSP Clock block is not used in a circuit, the default values of the DSP block are used.

6.4 PWM Generators

F2833x provides 6 sets of PWM outputs:

- PWM 1 (GPIO 0 and 1)
- PWM 2 (GPIO 2 and 3)
- PWM 3 (GPIO 4 and 5)
- PWM 4 (GPIO 6 and 7)
- PWM 5 (GPIO 8 and 9)
- PWM 6 (GPIO 10 and 11)

Each set has two outputs that are complementary to each other. For example PWM 1 has a positive output PWM 1A and a negative output PWM 1B, except when the PWM operates in a special operation mode.

In SimCoder, these 6 PWM's can be used in the following ways:

- Two 3-phase PWM generators: PWM 123 (consisting of PWM 1, 2, and 3) and PWM 456 (consisting of PWM 4, 5, and 6);
- Six 2-phase PWM generators: PWM 1, 2, 3, 4, 5, and 6, with the two outputs of each PWM generator not in a complementary way, but in special operation mode.
- 1-phase PWM generators: PWM 1, 2, 3, 4, 5, and 6, with two outputs complementary to each other.
- 1-phase PWM generators with phase shift: PWM 2, 3, 4, 5, and 6, with two outputs complementary to each other.

These PWM generators can trigger the A/D converter, and use trip-zone signals.

Beside the PWM generators described above, there are also 6 APWM generators that use the same resources as the captures. These PWM generators have restricted functionality as compared to the 6 PWM generators (PWM 1 to 6) as they can not trigger the A/D converter and can not use trip-zone signals. Also, because of the common resources, when a particular port is used for the capture, it can not be used for the PWM generator.

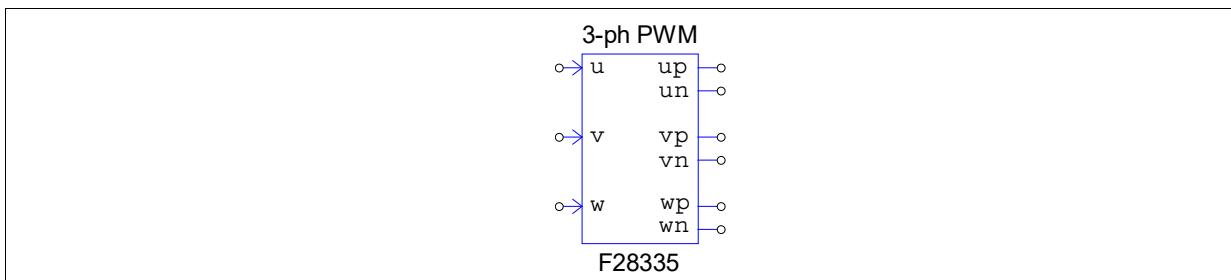
Note that all the PWM generators in SimCoder include one switching period delay internally. That is, the input value of a PWM generator is delayed by one cycle before it is used to update the PWM output. This delay is needed to simulate the delay inherent in the DSP hardware implementation.

PWM generators have a parameter called "PWM Freq. Scaling Factor". It can be set to 1 to 100. The hardware limit is 3. If the scaling factor is greater than 3, PWM will use an unused PWM to generator interrupt at the sampling frequency. This unused PWM is only used to generate a periodic interrupt, and its outputs can still be used for other functions. If there is no unused PWM in the system, a timer will be used.

6.4.1 3-Phase PWM

In the 3-phase PWM generator image, "u", "v", and "w" refer to the three phases (alternatively they are called Phase "a", "b", and "c"). The letter "p" refers to the positive output, and "n" refers to the negative output. For example, for 3-phase PWM 123, "up" is PWM1A, and "un" is PWM1B.

Image:



Attributes:

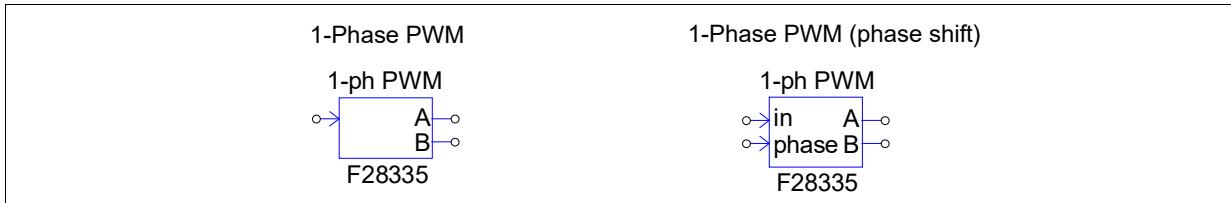
Parameters	Description
PWM Source	Source of the PWM generator. It can be either "3-phase PWM 123" that uses PWM 1 to 3, or "3-phase PWM 456" that uses PWM 4 to 6.
Dead Time	Dead time T_d for the PWM generator, in sec.
Sampling Frequency	Sampling frequency of the PWM generator, in Hz. The calculation is done and the PWM signal duty cycle is updated based on this frequency.
PWM Freq. Scaling Factor	The ratio between the PWM switching frequency and the sampling frequency. It can be from 1 to 100. For example, if the sampling frequency is 50 kHz and the scaling factor is 3, the PWM switching frequency will be 150 kHz. That is switches will operate at 150 kHz. But gating signals will be updated at 50 kHz, or once per 3 switching cycles.
Carrier Wave Type	Carrier wave type and the initial PWM output state. It can be one of the following: <ul style="list-style-type: none"> - <i>Triangular (start low)</i>: Triangular wave, and the initial PWM output state is low. - <i>Triangular (start high)</i>: Triangular wave, and the initial output state is high. - <i>Sawtooth (start low)</i>: Sawtooth wave, and the initial output state is low. - <i>Sawtooth (start high)</i>: Sawtooth wave, with the initial output state is high.
Trigger ADC	Setting whether for the PWM generator to trigger the A/D converter. It can be one of the following: <ul style="list-style-type: none"> - <i>Do not trigger ADC</i>: PWM does not trigger the A/D converter. - <i>Trigger ADC Group A</i>: PWM will trigger Group A of the A/D converter. - <i>Trigger ADC Group B</i>: PWM will trigger Group B of the A/D converter. - <i>Trigger ADC Group A&B</i>: PWM will trigger both Group A and B of the A/D converter.
ADC Trigger Position	A/D trigger position ranges from 0 to a value less than 1. When it is 0, the A/D converter is triggered at the beginning of the PWM cycle, and when it is 0.5, the A/D converter is triggered at the 180° position of the PWM cycle.
Use Trip-Zone i	Define whether the PWM generator uses the i_{th} trip-zone signal or not, where i ranges from 1 to 6. It can be one of the following: <ul style="list-style-type: none"> - <i>Disable Trip-Zone i</i>: Disable the i_{th} trip-zone signal. - <i>One shot</i>: The PWM generator uses the trip-zone signal in the one-shot mode. Once triggered, the PWM must be started manually. - <i>Cycle by cycle</i>: The PWM generator uses the trip-zone signal in the cycle-by-cycle basis. The trip-zone signal is effective within the current cycle, and PWM will automatically re-start in the next cycle.
Trip Action	Define how the PWM generator responds to the trip action. It can be one of the following: <ul style="list-style-type: none"> - <i>High impedance</i>: The PWM outputs are in high impedance. - <i>PWM A high & B low</i>: The PWM positive output is high, and the negative output is low. - <i>PWM A low & B high</i>: The PWM positive output is low, and the negative output is high. - <i>No action</i>: No action is taken.
Peak-to-Peak Value	Peak-to-peak value V_{pp} of the carrier wave
Offset Value	DC offset value V_{offset} of the carrier wave

Initial Input Value u, v, w	Initial value of 3-phase inputs u , v , and w
Start PWM at Beginning	When it is set to <i>Start</i> , PWM will start right from the beginning. If it is set to <i>Do not start</i> , one needs to start PWM using the Start PWM block.
Simulation Output Mode	<p>The simulation output mode can be set to <i>Switching mode</i> or <i>Average mode</i>. When it is set to "Switching mode", the outputs of the PWM block are PWM signals. When it is set to "Average mode", the outputs of the PWM block are average mode signals.</p> <p>In the average mode, if the carrier wave is from negative to positive, and the absolute values of the negative peak and the positive peak are equal (for example, the carrier wave is from -1 to +1, or from -5 to +5), the modulation is considered as an ac signal modulation. Otherwise, the modulation is considered as a dc signal modulation. For example, modulation in a 3-phase or single-phase inverter is an ac modulation, and modulation in a buck converter is a dc modulation.</p> <p>In the ac signal modulation, if the input u of the PWM block is V_u, the output up and un in average mode will be:</p> $V_{up} = V_u / (V_{pp} + V_{offset})$ $V_{un} = -V_{up}$ <p>In this case, V_u is between $-(V_{pp} + V_{offset})$ and $V_{pp} + V_{offset}$, and V_{up} is between -1 to +1.</p> <p>In the dc signal modulation, the output up and un in average mode will be:</p> $V_{up} = (V_u - V_{offset}) / V_{pp}$ $V_{un} = 1 - V_{up}$ <p>In this case, V_u is between V_{offset} and $V_{pp} + V_{offset}$, and V_{up} is between 0 to +1.</p> <p>When it is set to the average mode, the PWM block outputs can be connected to a converter/inverter in the average mode model.</p>

6.4.2 1-Phase PWM and 1-Phase PWM (phase shift)

The attributes for the **1-Phase PWM** block and **1-phase PWM (phase shift)** block are mostly the same. The difference is that the 1-Phase PWM block defines the phase shift through a parameter, while the 1-Phase PWM (phase shift) block reads the phase shift from an external input (labeled as "phase" in the image). The phase shift is in degree.

Images:



Attributes:

Parameters	Description
PWM Source	Source of the PWM generator. Without phase shift, it can be PWM 1 to PWM 6. With phase shift, it can be PWM 2 to PWM 6.
Output Mode	Output mode of the PWM generator. It can be one of the following: <ul style="list-style-type: none"> - <i>Use PWM A&B</i>: Both PWM outputs A and B are used, and they are complementary. - <i>Use PWM A</i>: Only PWM output A is used. - <i>Use PWM B</i>: Only PWM output B is used.
Dead Time	Dead time T_d for the PWM generator, in sec.
Sampling Frequency	Sampling frequency of the PWM generator, in Hz. The calculation is done and the PWM signal duty cycle is updated based on this frequency.
PWM Freq. Scaling Factor	The ratio between the PWM switching frequency and the sampling frequency. It can be from 1 to 100. For example, if the sampling frequency is 50 kHz and the scaling factor is 3, the PWM switching frequency will be 150 kHz. That is switches will operate at 150 kHz. But gating signals will be updated at 50 kHz, or once per 3 switching cycles.
Carrier Wave Type	Carrier wave type and the initial PWM output state. It can be one of the following: <ul style="list-style-type: none"> - <i>Triangular (start low)</i>: Triangular wave, and the initial PWM output state is low. - <i>Triangular (start high)</i>: Triangular wave, and the initial output state is high. - <i>Sawtooth (start low)</i>: Sawtooth wave, and the initial output state is low. - <i>Sawtooth (start high)</i>: Sawtooth wave, and the initial output state is high.
Trigger ADC	Setting whether for the PWM generator to trigger the A/D converter. It can be one of the following: <ul style="list-style-type: none"> - <i>Do not trigger ADC</i>: PWM does not trigger the A/D converter. - <i>Trigger ADC Group A</i>: PWM will trigger Group A of the A/D converter. - <i>Trigger ADC Group B</i>: PWM will trigger Group B of the A/D converter. - <i>Trigger ADC Group A&B</i>: PWM will trigger both Group A and B of the A/D converter.
ADC Trigger Position	A/D trigger position ranges from 0 to a value less than 1. When it is 0, the A/D converter is triggered at the beginning of the PWM cycle, and when it is 0.5, the A/D converter is triggered at the 180° position of the PWM cycle.
Use Trip-Zone i	Define whether the PWM generator uses the i_{th} trip-zone signal or not, where i ranges from 1 to 6. It can be one of the following: <ul style="list-style-type: none"> - <i>Disable Trip-Zone i</i>: Disable the i_{th} trip-zone signal. - <i>One shot</i>: The PWM generator uses the trip-zone signal in the one-shot mode. Once triggered, the PWM must be started manually. - <i>Cycle by cycle</i>: The PWM generator uses the trip-zone signal in the cycle-by-cycle basis. The trip-zone signal is effective within the current cycle, and PWM will automatically re-start in the next cycle.
Trip Action	Define how the PWM generator responds to the trip action. It can be one of the following: <ul style="list-style-type: none"> - <i>High impedance</i>: The PWM outputs are in high impedance. - <i>PWM A high & B low</i>: The PWM positive output is high, and the negative output is low. - <i>PWM A low & B high</i>: The PWM positive output is low, and the negative output is high. - <i>No action</i>: No action is taken.

Peak-to-Peak Value	Peak-to-peak value V_{pp} of the carrier wave
Offset Value	DC offset value V_{offset} of the carrier wave
Phase Shift	Phase shift of the output with respect to the reference PWM generator output, in degree. Note that this parameter is for 1-phase PWM only. For 1-phase PWM (phase shift) , the phase shift is read from an external input.
Initial Input Value	Initial value of the input
Use HRPWM	<p>Define the high-resolution PWM. It can be one of the following:</p> <ul style="list-style-type: none"> - <i>Do not use HRPWM</i>: Do not use high-resolution PWM - <i>Use HRPWM without calibration</i>: Use high-resolution PWM without calibration - <i>Use HRPWM with calibration</i>: Use high-resolution PWM with calibration
Start PWM at Beginning	When it is set to <i>Start</i> , PWM will start right from the beginning. If it is set to <i>Do not start</i> , one needs to start PWM using the Start PWM block.
Simulation Output Mode	<p>The simulation output mode can be set to <i>Switching mode</i> or <i>Average mode</i>.</p> <p>When it is set to "Switching mode", the outputs of the PWM block are PWM signals. When it is set to "Average mode", the outputs of the PWM block are average mode signals.</p> <p>In the average mode, if the carrier wave is from negative to positive, and the absolute values of the negative peak and the positive peak are equal (for example, the carrier wave is from -1 to +1, or from -5 to +5), the modulation is considered as an ac signal modulation. Otherwise, the modulation is considered as a dc signal modulation. For example, modulation in a 3-phase or single-phase inverter is an ac modulation, and modulation in a buck converter is a dc modulation.</p> <p>In the ac signal modulation, if the input of the PWM block is V_{in}, the output A and B in average mode will be:</p> $V_A = V_{in} / (V_{pp} + V_{offset})$ $V_B = -V_A$ <p>In this case, V_{in} is between $-(V_{pp} + V_{offset})$ and $V_{pp} + V_{offset}$, and V_A and V_B are between -1 to +1.</p> <p>In the dc signal modulation, the output A and B in average mode will be:</p> $V_A = (V_{in} - V_{offset}) / V_{pp}$ $V_B = 1 - V_A$ <p>In this case, V_{in} is between V_{offset} and $V_{pp} + V_{offset}$, and V_A and V_B are between 0 to +1.</p> <p>When it is set to the average mode, the PWM block outputs can be connected to a converter/inverter in the average mode model.</p>

Phase Shift

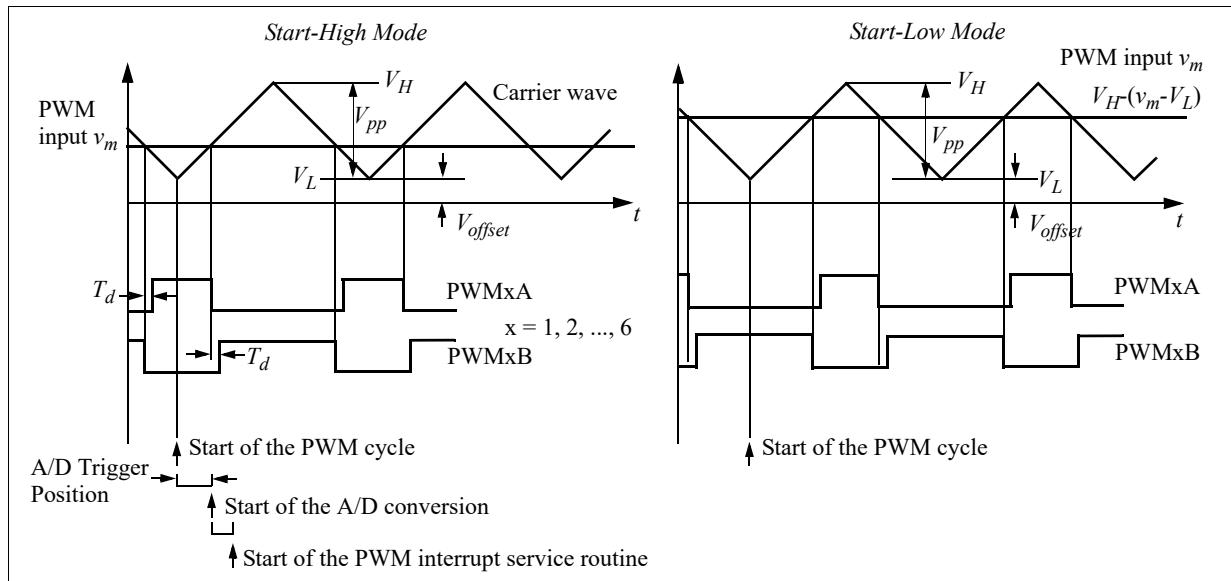
A 1-phase PWM block can generate PWM signal that is phase shifted with respect to another PWM signal. The way how PWM blocks are defined for phase shift is explained in Section 6.4.5.

The phase shift value is in degree. When the value is -30° , the output will be shifted to the right (lagging) by 30° of the switching cycle with respect to the reference PWM generator output. This is equivalent to shifting the PWM carrier wave to the right by 30° . When the phase value is 30° , the output will be shifted to the left (leading) by 30° .

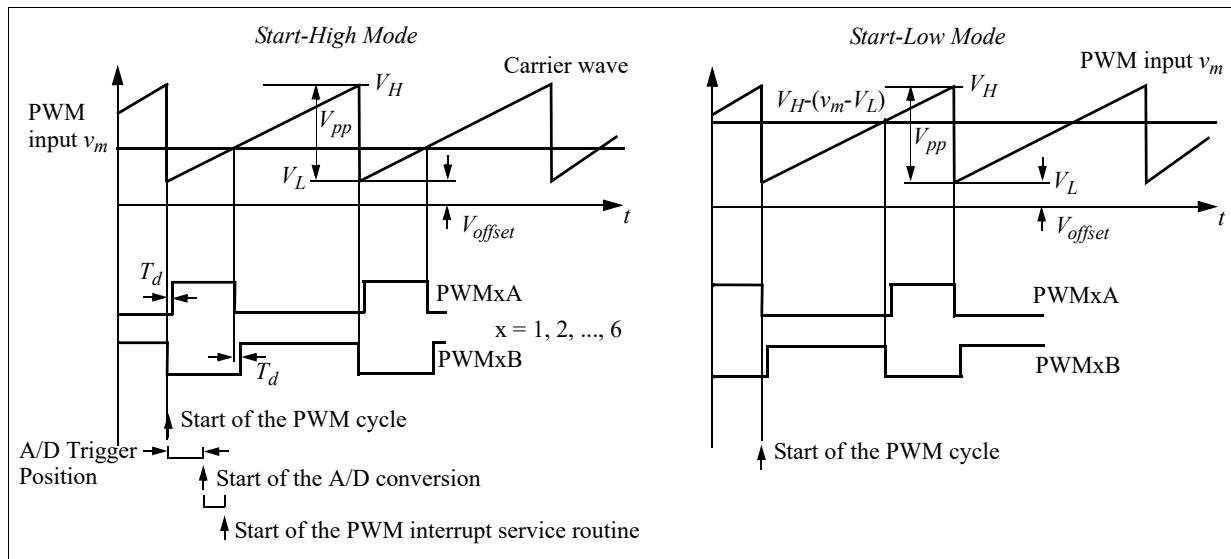
Carrier Wave

There are two types of carrier waveforms: triangular wave (with equal rising and falling slope intervals) and

sawtooth wave. In addition, there are two operation modes: start-low and start-high modes, as explained below. The input and output waveforms of a PWM generator with the triangular carrier wave are shown below:



The input and output waveforms of a PWM generator with the sawtooth carrier wave are shown below:



The figures above show how the dead time is defined, and the time sequence when the PWM generator triggers the A/D converter. If triggering the A/D converter is selected, from the start of the PWM cycle, after a certain delay defined by the A/D trigger position, the A/D conversion will start. After the A/D conversion is completed, the PWM interrupt service routine will start.

If the PWM generator does not trigger the A/D converter, the PWM interrupt service routine will start at the beginning of the PWM cycle.

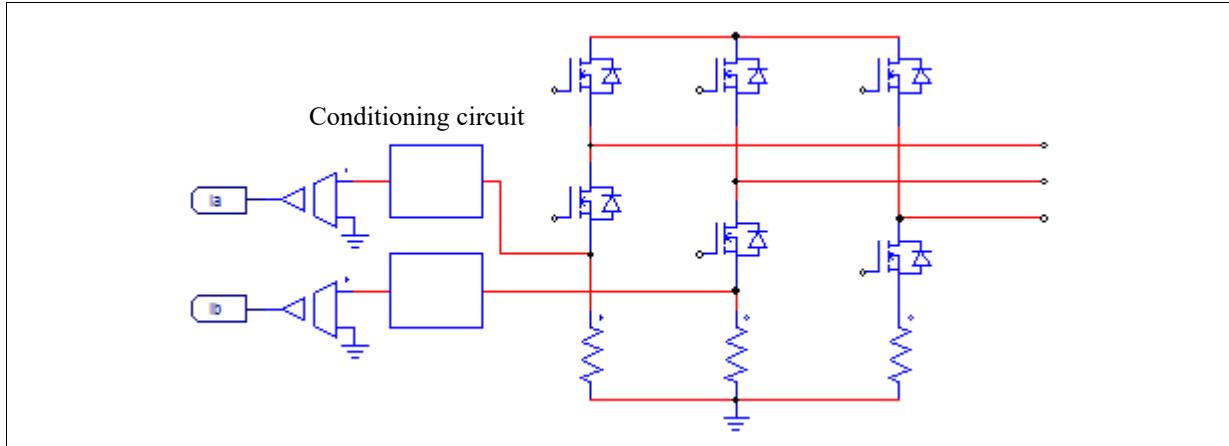
The figures above also show how the start-high and start-low modes work. Assume that the PWM input is v_m , and the lowest value of the carrier wave is V_L and the highest value is V_H . In the start-high mode, the PWM positive output PWMA is high at the beginning of the switching cycle, and it remains high as long as the input v_m is greater than the carrier wave. For example, for a carrier wave from 0 to 1, $V_L=0$, and $V_H=1$. If $v_m=0.2$, the PWM output PWMA will remain high as long as the carrier is less than 0.2.

On the other hand, in the start-low mode, the PWM positive output PWMA is low at the beginning of the switching cycle, and it is high when the carrier wave is greater than the value $V_H - (v_m - V_L)$. For example, for a

carrier wave from 0 to 1, $V_L=0$, and $V_H=1$. If $v_m=0.2$, the PWM output PWMA will be high as long as the carrier is greater than 0.8.

The carrier start mode depends on how switch currents are measured. In a 3-phase inverter, for example, if top switch currents are measured, the start-high mode should be selected. This ensures that at the beginning of the cycle, the top switch gating signal is high and the current is conducting. On the other hand, if bottom switch currents are measured, the start-low mode should be selected. This ensures that at the beginning of the cycle, the top switch gating signal is low, and the bottom switch gating signal is high and the current is conducting.

For example, in the circuit below, the bottom switch currents of Phase A and B are measured. In this case, the carrier start-low mode should be selected.

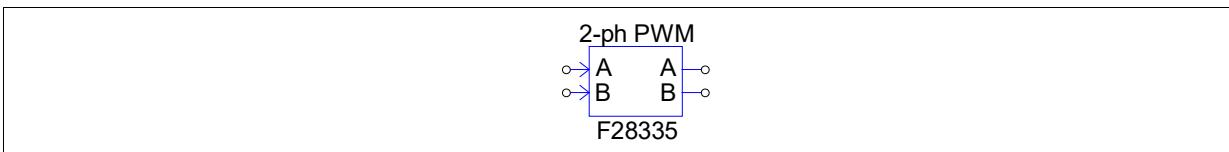


Note: In the start-low mode, the PWM input v_m is converted to $V_H(v_m-V_L)$ internally before it is compared with the carrier wave to generate the PWM signal. With the conversion, both the start-low and start-high modes will have the same duty cycle expression. For example, for a sawtooth wave with $V_L=0$ and $V_H=1$, or for a triangular wave with $V_L= -V_H$, the duty cycle D of the PWMA output in both the start-low and start-high modes is: $D = v_m/V_H$.

6.4.3 2-Phase PWM

A 2-phase PWM block operates in one of 6 operation modes.

Image:



Attributes:

Parameters	Description
PWM Source	Source of the PWM generator. It can be PWM 1 to PWM 6.
Mode Type	Operation mode of the PWM generation. It can be one of the 6 modes. The waveforms of the 6 operation modes are described below.
Sampling Frequency	Sampling frequency of the PWM generator, in Hz. The calculation is done and the PWM signal duty cycle is updated based on this frequency.
PWM Freq. Scaling Factor	The ratio between the PWM switching frequency and the sampling frequency. It can be from 1 to 100. For example, if the sampling frequency is 50 kHz and the scaling factor is 3, the PWM switching frequency will be 150 kHz. That is switches will operate at 150 kHz. But gating signals will be updated at 50 kHz, or once per 3 switching cycles.

Trigger ADC	Setting whether for the PWM generator to trigger the A/D converter. It can be one of the following: <ul style="list-style-type: none"> - <i>Do not trigger ADC</i>: PWM does not trigger the A/D converter. - <i>Trigger ADC Group A</i>: PWM will trigger Group A of the A/D converter. - <i>Trigger ADC Group B</i>: PWM will trigger Group B of the A/D converter. - <i>Trigger ADC Group A&B</i>: PWM will trigger both Group A and B of the A/D converter.
ADC Trigger Position	A/D trigger position ranges from 0 to a value less than 1. When it is 0, the A/D converter is triggered at the beginning of the PWM cycle, and when it is 0.5, the A/D converter is triggered at the 180° position of the PWM cycle.
Use Trip-Zone <i>i</i>	Define whether the PWM generator uses the i_{th} trip-zone signal or not, where i ranges from 1 to 6. It can be one of the following: <ul style="list-style-type: none"> - <i>Disable Trip-Zone i</i>: Disable the i_{th} trip-zone signal. - <i>One shot</i>: The PWM generator uses the trip-zone signal in the one-shot mode. Once triggered, the PWM must be started manually. - <i>Cycle by cycle</i>: The PWM generator uses the trip-zone signal in the cycle-by-cycle basis. The trip-zone signal is effective within the current cycle, and PWM will automatically re-start in the next cycle.
Trip Action	Define how the PWM generator responds to the trip action. It can be one of the following: <ul style="list-style-type: none"> - <i>High impedance</i>: The PWM outputs are in high impedance. - <i>PWM A high & B low</i>: The positive output of the PWM is high, and the negative output is low. - <i>PWM A low & B high</i>: The positive output of the PWM is low, and the negative output is high. - <i>No action</i>: No action is taken.
Peak Value	Peak value V_{pk} of the carrier wave
Initial Input Value A, B	Initial value of the inputs A and B.
Start PWM at Beginning	When it is set to <i>Start</i> , PWM will start right from the beginning. If it is set to <i>Do not start</i> , one needs to start PWM using the Start PWM block.

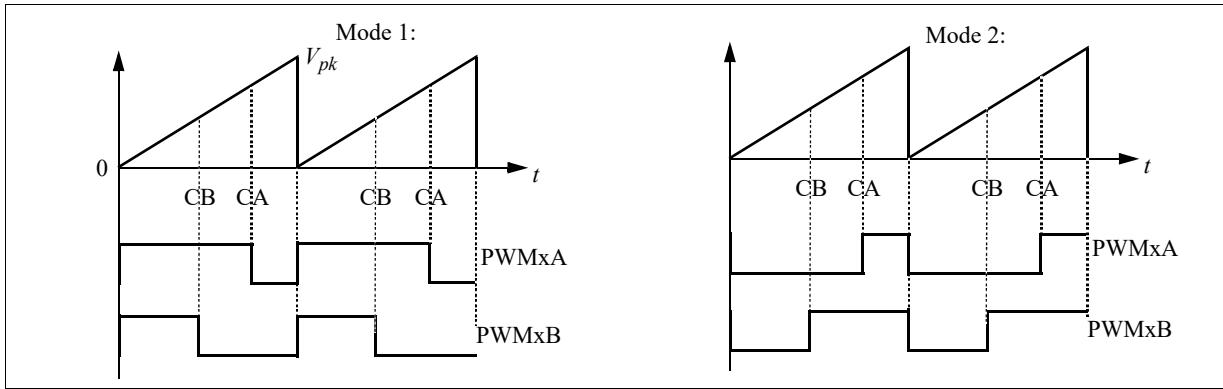
For 2-phase PWM generators, the outputs are determined based on the mode of operation, as described below. The carrier wave is either sawtooth or triangular, depending on the mode of operation. It increases from 0 to the peak value V_{pk} , and there is no dc offset.

Operation Mode 1:

The figure below on the left shows the waveforms of Mode 1. In the figure, "CA" and "CB" refer to two inputs A and B of the 2-phase PWM generator. Each input controls the turn-off time of each output.

Operation Mode 2:

The figure below on the right shows the waveforms of Mode 2. Unlike in Mode 1, each input controls the turn-on time of each output.

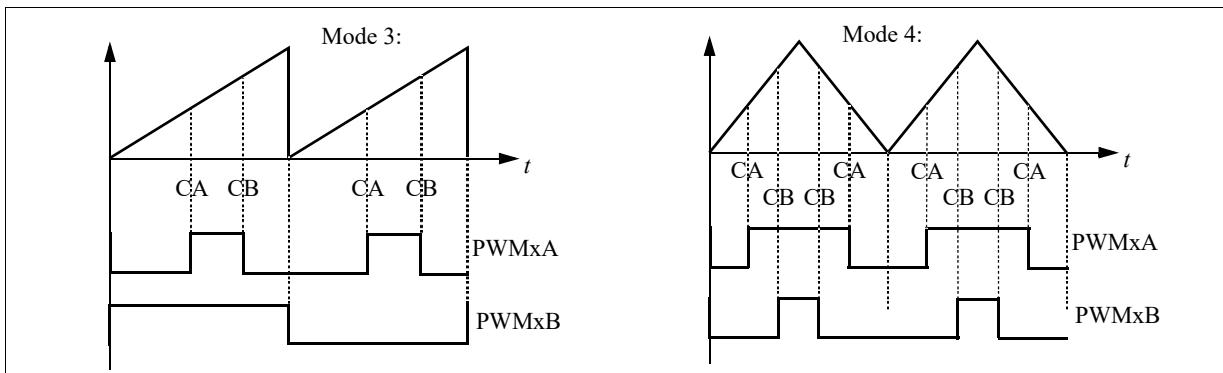


Operation Mode 3:

The figure below on the left shows the waveforms of Mode 3. Input A controls the turn-on and Input B controls the turn-off of the PWM output A. The PWM output B is on for one complete PWM cycle, and is off for the next cycle.

Operation Mode 4:

The figure below on the right shows the waveforms of Mode 4. The carrier wave is triangular. Each input controls both the turn-on and turn-off of its output.

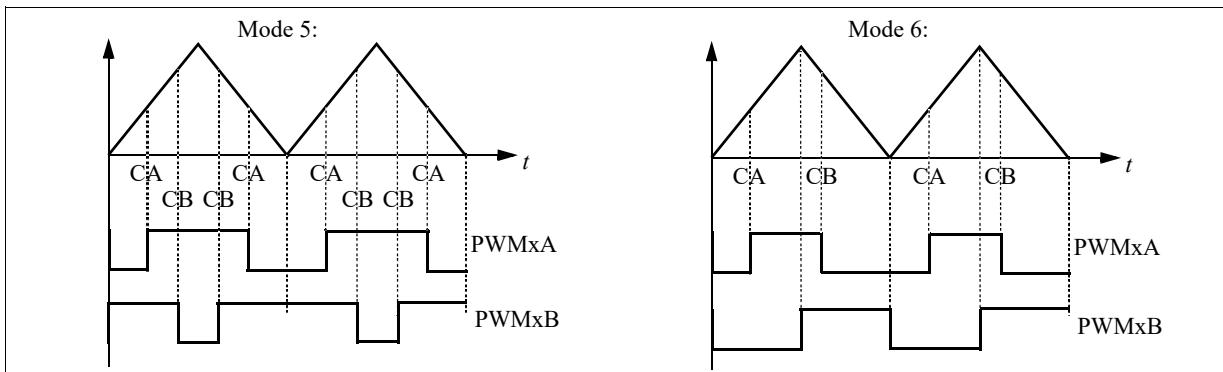


Operation Mode 5:

The figure below on the left shows the waveforms of Mode 5. The carrier wave is triangular. Similar to Mode 4, each input controls both the turn-on and turn-off of its output. Note that PWM output B is inverted in this case.

Operation Mode 6:

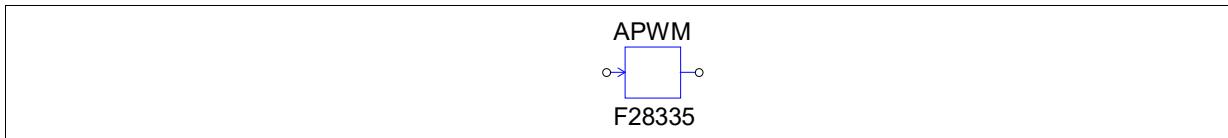
The figure below on the right shows the waveforms of Mode 6. Input A controls the turn-on and Input B controls the turn-off of PWM output A. The PWM output B is on for the first half PWM cycle, and is off for the second half cycle.



6.4.4 Single PWM (shared with capture)

A single PWM generator, also called APWM, shares the same resource as captures.

Image:



Attributes:

Parameters	Description
PWM Source	The PWM source can be one of the six APWM's in 14 designated GPIO ports, as listed below: <ul style="list-style-type: none">- APWM 1 (GPIO5, 24, 34)- APWM 2 (GPIO7, 25, 37)- APWM 3 (GPIO9, 26)- APWM 4 (GPIO11, 27)- APWM 5 (GPIO3, 48)- APWM 6 (GPIO1, 49)
PWM Frequency	Frequency of the PWM generator, in Hz
Carrier Wave Type	Carrier wave type and the initial PWM output state. It can be one of the following: <ul style="list-style-type: none">- <i>Sawtooth (start low)</i>: sawtooth wave, with the PWM output low initially.- <i>Sawtooth (start high)</i>: Sawtooth wave, with the PWM output high initially.
Stop Action	Output status when the PWM generator is stopped. It can be one of the following: <ul style="list-style-type: none">- Output low: The PWM output will be set to low.- Output high: The PWM output will be set to high.
Peak-to-Peak Value	Peak-to-peak value of the carrier wave
Offset Value	DC offset value of the carrier wave
Phase Shift	Phase shift of the output with respect to the reference PWM generator, in deg.
Initial Input Value	Initial value of the input
Start PWM at Beginning	When it is set to <i>Start</i> , PWM will start right from the beginning. If it is set to <i>Do not start</i> , one needs to start PWM using the Start PWM block.

Similar to 1-phase PWM generators, an APWM generator can generate a PWM signal that has a phase shift with respect to another PWM generator. The way how PWM blocks are defined for phase shift is explained in Section 6.4.5.

The phase shift value is in degree. When the value is -30° , the output will be shifted to the right (lagging) by 30° of the switching cycle with respect to the reference PWM generator output. This is equivalent to shifting the PWM carrier wave to the right by 30° . When the phase value is 30° , the output will be shifted to the left (leading) by 30° .

As noted before, the APWM generators has reduced number of functions than 1-phase PWM generators. It can not trigger the A/D converter and can not use the trip-zone signal.

6.4.5 Synchronization Between PWM Blocks

Three types of PWM blocks can be synchronized, and phase shifts can be defined between each other: **1-phase PWM**, **1-phase PWM (phase shift)**, and **APWM (or Single PWM (shared with capture))**.

A 1-phase PWM block can generate PWM signal that is phase shifted with respect to another PWM signal. There are two series in regular PWM blocks: PWM 1, 2, 3; and PWM 1, 4, 5, 6.

Similarly, there are two series in APWM blocks for phase shift: PWM 1, APWM 1, 2, 3; and PWM 1, APWM 4, 5, 6.

The definitions of the PWM blocks for phase shift are described below.

- The reference PWM and the PWM being phase shifted must be from the same series. That is, PWM 1 can be the reference, and PWM 2 and 3, or PWM 4, 5, and 6, can be phase shifted with respect to PWM 1. Or PWM 2 can be the reference, and PWM 3 can be phase shifted with respect to PWM 2.

Similarly, PWM 4 (or 5) can be the reference, and PWM 5 (or 6) can be phase shifted with respect to PWM 4 (or 5). But using PWM 2 or 3 as the reference for PWM 4, 5, or 6 is not allowed.

This also applies to APWMs. For example, PWM 1 can be the reference, and APWM 1, 2, and 3 can be phase shifted. Or APWM 1 can be the reference, and APWM 2 and 3 can be phase shifted. Similarly, PWM 1 can be the reference, and APWM 4, 5, and 6 can be phase shifted. Or APWM 4 can be the reference, and APWM 5 and 6 can be phase shifted.

- The reference PWM and the PWM being shifted must be consecutive in the series, unless the skipped PWM is not used. For example, if PWM 1, 2, and 3 are all used, but PWM 2 is not synchronized with PWM 1 and 3, this is not allowed. However, if PWM 2 is not used in the circuit, it is ok to use PWM 1 as the reference and phase shift PWM 3.

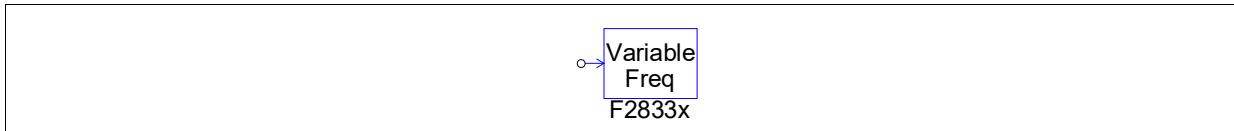
For example, the following definitions are correct, with the first PWM as the reference and the subsequent PWM blocks phase shifted:

PWM 1 (reference), PWM 2, PWM 3
PWM 1 (reference), PWM 4, PWM 5, PWM 6
PWM 2 (reference), PWM 3
PWM 4 (reference), PWM 5
PWM 5 (reference), PWM 6
PWM 1 (reference), APWM 1, APWM 2, APWM 3
PWM 1 (reference), APWM 4, APWM 5, APWM 6
APWM 1 (reference), APWM2, APWM 3
APWM 4 (reference), APWM5, APWM 6

6.5 Variable Frequency PWM

The Variable Frequency PWM block provides the function to change the sampling frequency of a PWM generator. The image and parameters are shown below.

Image:



Attributes:

Parameters	Description
PWM Source	Source of the PWM generator. It can be one of the following: PWM 1, PWM 2, PWM 3, PWM 4, PWM 5, PWM 6, 3-phase PWM 123, and 3-phase PWM 456.
Adjust Interrupt Pos.	Specify if the interrupt position is adjusted with the frequency. It can be one of the following: - <i>Do not adjust</i> : The interrupt position will remain unchanged as calculated with the base frequency. - <i>Adjust</i> : The interrupt position will be recalculated at the beginning of each cycle based on the new frequency.

The sampling frequency of the corresponding PWM block will be changed at the beginning of the next PWM period as follows:

$$\text{PWM_Frequency} = \text{PWM_Base Frequency} / \text{Input_Value}$$

where *PWM_Base_Frequency* is the sampling frequency of the corresponding PWM block, and *Input_Value* is the input value of this block.

If the interrupt position is to be adjusted, the interrupt position will be recalculated in each cycle. Since adjusting the interrupt position takes time, if the frequency change is small, it is recommended not to adjust the interrupt position.

6.6 Start PWM and Stop PWM

The Start PWM and Stop PWM blocks provide the function to start/stop a PWM generator. The images and parameters are shown below.

Image:



Attributes:

Parameters	Description
PWM Source	Source of the PWM generator. It can be: PWM 1-6, 3-phase PWM 123 and PWM 456, and Capture 1-6.

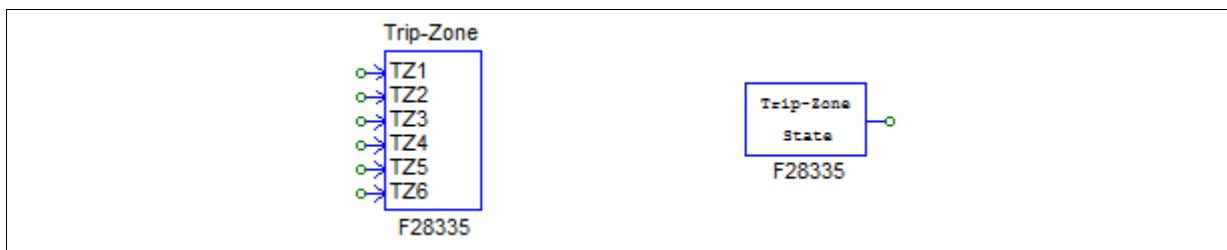
6.7 Trip-Zone and Trip-Zone State

F2833x provides 6 trip-zones, Trip-Zone 1 to 6 which use the ports GPIO12 to GPIO17. Trip-zone is used to handle external fault or trip conditions. The corresponding PWM outputs can be programmed to respond accordingly.

One trip-zone signal can be used by multiple PWM generators, and a PWM generator can use any or all of the 6 trip-zone signals. The interrupt generated by trip-zone signals are handled by the interrupt block.

The trip-zone signal triggers a trip action when the input signal is low (0).

Image:



Attributes for Trip-Zone:

Parameters	Description
Port GPIO12 as Trip-Zone 1	Define if Port GPIO12 is used as trip-zone 1.
Port GPIO13 as Trip-Zone 2	Define if Port GPIO13 is used as trip-zone 2.
Port GPIO14 as Trip-Zone 3	Define if Port GPIO14 is used as trip-zone 3.
Port GPIO15 as Trip-Zone 4	Define if Port GPIO15 is used as trip-zone 4.

Port GPIO16 as Trip-Zone 5	Define if Port GPIO16 is used as trip-zone 5.
Port GPIO17 as Trip-Zone 6	Define if Port GPIO17 is used as trip-zone 6.

Attributes for Trip-Zone State:

Parameters	Description
PWM Source	Source of the PWM generator. It can be: PWM 1-6, and 3-phase PWM 123 and PWM 456.

The trip-zone interrupt can be generated in either one-shot mode or cycle-by-cycle mode, as defined in the PWM generator parameter input. In the cycle-by-cycle mode, the interrupt only affects the PWM output within the current PWM cycle. On the other hand, in the one-shot mode, interrupt triggers a trip action when the input signal is low (0). will set the PWM output permanently, and the PWM generator must be restarted to resume the operation.

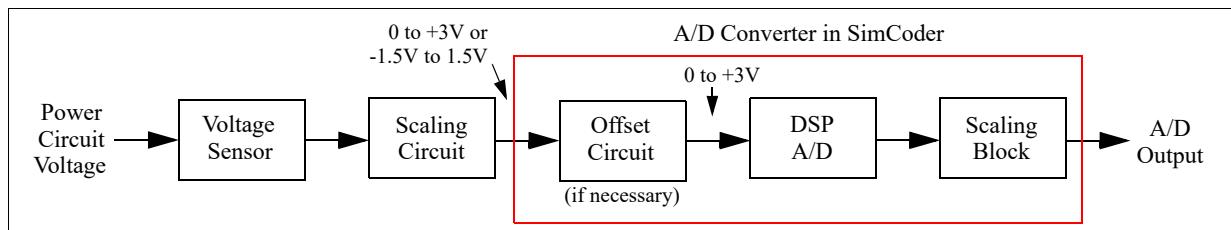
The Trip-Zone State element indicates whether the trip-zone signal is in one-shot mode or cycle-by-cycle mode when it triggers a PWM generator to generate an interrupt. When the output is 1, it means that the trip-zone signal is in one-shot mode. When the output is 0, the trip-zone signal is in cycle-by-cycle mode.

Note that when defining the interrupt block associate with trip-zone, the "Device Name" parameter of the interrupt block should be the name of the PWM generator, not the trip-zone block name. For example, if a PWM generator called "PWM_G1" uses trip-zone 1 in the trip-zone block "TZ1". The "Device Name" of the corresponding interrupt block should be "PWM_G1", not "TZ1". The "Channel Number" parameter in the interrupt block is not used in this case.

6.8 A/D Converter

F2833x provides a 12-bit 16-channel A/D converter. It is divided into two groups: Group A and Group B. The input range of the physical A/D converter on the DSP is from 0V to +3V.

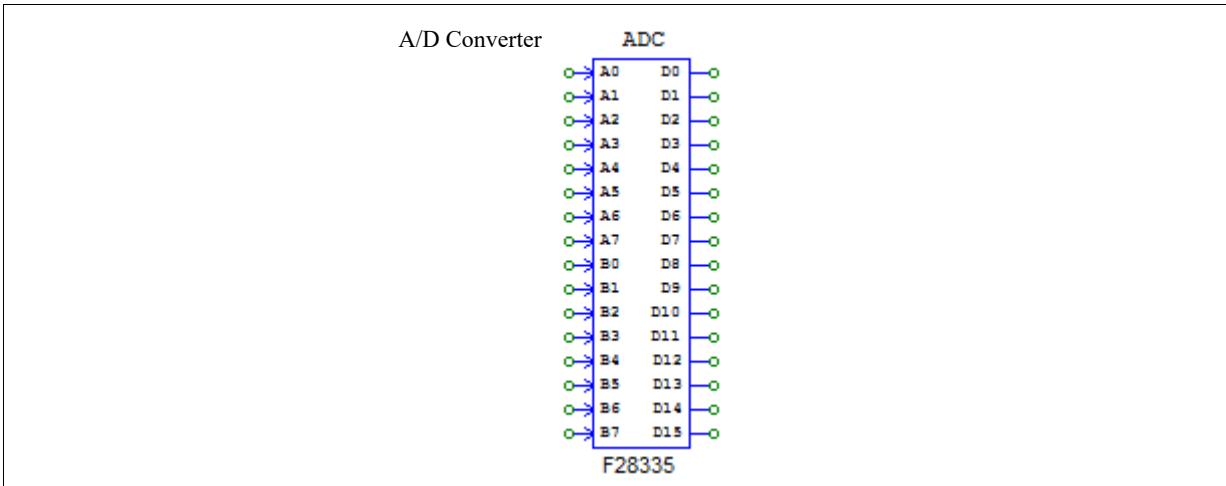
Normally a power circuit quantity (voltage, current, speed, etc.) is brought to the DSP in several stages. For example, a power circuit voltage, which could be at a high level, is first converted to a control signal using a voltage sensor. A scaling circuit is then used to scale the signal, and an offset circuit is used to provide dc offset to the signal if necessary, so that the signal at the DSP A/D input is within the 0V and +3V. This signal is converted to a digital value in DSP, and a scaling block may be used to scale the value back to its original value. The complete process is shown in the diagram below.



As shown above, the A/D converter element in SimCoder is not exactly the same as the physical A/D converter on the DSP. Rather, it combines the functions of an offset circuit, the DSP A/D converter, and a scaling block. This is designed for the convenience of AC system applications.

The image and the parameters of the A/D converter in the SimCoder library are described below. In the following description, "A/D converter" refers to the A/D converter in the SimCoder library, not the DSP A/D converter, unless otherwise stated.

Image:



Attributes:

Parameters	Description
ADC Mode	Mode of operation of the A/D converter. It can be one of the following: <ul style="list-style-type: none"> - <i>Continuous</i>: The A/D converter performs the conversion continuously. When the converter value is read, the result of the last conversion is read. - <i>Start/stop (8-channel)</i>: The A/D converter only performs the conversion upon request, on only one of the 8-channel groups. - <i>Start/stop (16-channel)</i>: The A/D converter only performs the conversion upon request, on all 16 channels.
Ch A_i or B_i Mode	Input mode of the A/D converter channel A_i or B_i , where i is from 0 to 7. The input mode can be one of the following: <ul style="list-style-type: none"> - <i>AC</i>: This option is for simulation only, not for code generation. The input range is considered from -1.5V to +1.5V. This option includes the offset circuit into the A/D converter. It provides the convenience in cases where an external level shifter is needed to shift the AC signal to the 0 to +3V range. - <i>DC</i>: The input is a dc value, and the range is from 0 to +3V.
Ch A_i or B_i Gain	Gain k of the A/D converter channel A_i (or B_i), where i is from 0 to 7.

Mode of Operation:

The A/D converter can perform conversion autonomously when it is set to the "Continuous" mode. The "Start/Stop" mode is for the conversion to be triggered by a PWM generator.

Note the following restrictions in using PWM generator triggered A/D converter:

- The A/D converter can be triggered by only one PWM generator. That is, if there are multiple PWM generators, only one can be set to trigger the A/D converter, and the rest should be set not to trigger the A/D converter.
- It is not permitted to have the A/D converter triggered by one PWM generator while some of the signals in this group are also used in a circuit that has a different sampling rate than the frequency of the PWM generator.

In these situations, it is recommended that the A/D converter be set to the "Continuous" mode.

Output Scaling:

The output is scaled based on the following:

$$V_o = k * V_i$$

where V_i is the value at the input port of the A/D converter.

Input Offset and Scaling:

Note that the input of the A/D converter must stay within the input range. When the input is out of the range, it will be clamped to the limit, and a warning message will be given.

Also, the signal at the input port of the A/D converter must be scaled such that, when the input channel mode is DC, the maximum input voltage be scaled to +3V; and when the input channel mode is AC, the maximum peak voltage be scaled to +1.5V.

In many applications, the circuit variables to be monitored are AC signals, especially in AC motor drive systems. For each of this kind of AC signals, an offset circuit must be built in the hardware on circuit board at the input of the DSP analog input, in order to shift the signal level to the acceptable range of 0 to +3.0V.

SimCoder's A/D converter for F2833x DSP provides the convenience for such cases. Instead of level-shifting and scaling the A/D output signals, user may chose to use the offset option and scaling factor in the SimCoder A/D converter, and the target code will be generated accordingly.

To illustrate how to use the A/D converter, two examples are given below: One with a dc input and the other with an ac input.

A/D Converter Channel DC Mode Example

Assume that a power circuit voltage is a dc quantity, and the range is as follows:

$$V_{i_min} = 0 \text{ V}, V_{i_max} = 150 \text{ V}$$

The input mode of the A/D converter will be set to dc, and the input range is from 0 to +3V. Assume that the actual value of the voltage at a certain point is:

$$V_i = 100 \text{ V}$$

Let the voltage sensor gain be 0.01. After the voltage sensor, the maximum value and the actual value of the input become:

$$V_{i_max_s} = 150 * 0.01 = 1.5 \text{ V}$$

$$V_{i_s} = 100 * 0.01 = 1 \text{ V}$$

To utilize the full range of the DSP, a conditioning circuit with a gain of 2 will be used. The combined gain of the voltage sensor and the conditioning circuit becomes: $0.01 * 2 = 0.02$. After the conditioning circuit and at the input of the DSP A/D converter, the maximum value and the actual value of the input become:

$$V_{i_max_s_c} = 1.5 * 2 = 3 \text{ V}$$

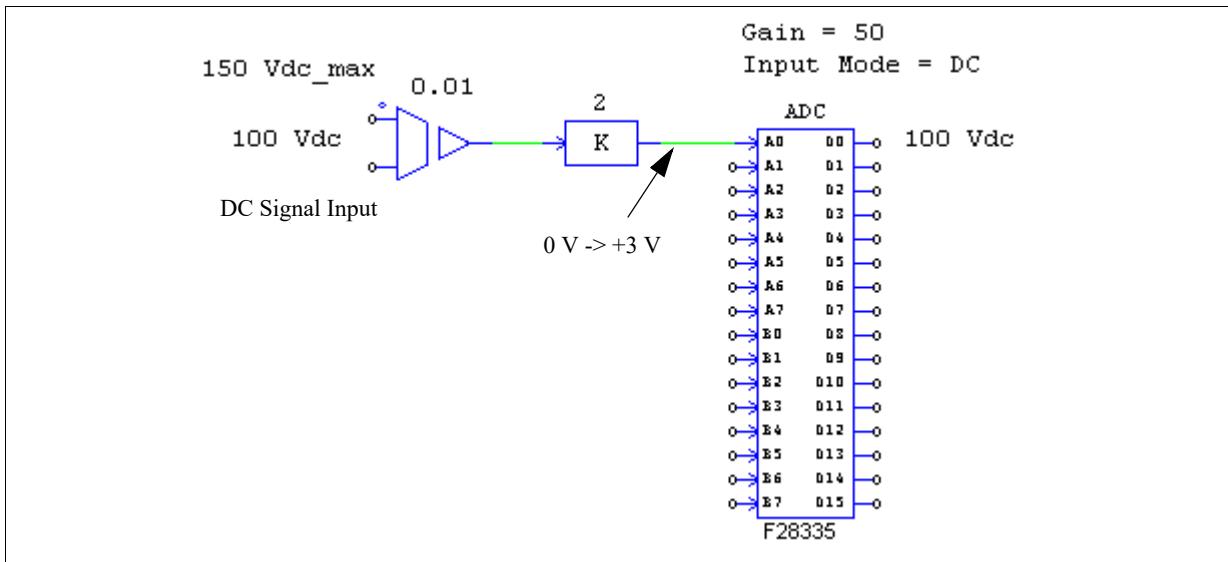
$$V_{i_s_c} = 1 * 2 = 2 \text{ V}$$

The scaling block after the DSP A/D can be selected such that the original power circuit quantity is restored. In this example, a gain of 50 will be used. Note that this is the reciprocal of the combined gain of the voltage sensor and the conditioning circuit. At the A/D output, the maximum value and the actual value are:

$$V_{o_max} = 50 * 3 = 150 \text{ V}$$

$$V_o = 50 * 2 = 100 \text{ V}$$

The gain of the A/D channel in PSIM will be set to 50. The circuit connection and the settings are shown in the figure below.



Please note that, in this example, if the gain of the proportional block is changed from 2 to 1, and the A/D gain is changed from 50 to 100, the simulation results will be the same. But the generated hardware code will not be correct. This is because the hardware code assumes that the maximum input value is scaled to +3V, but in this case it is only +1.5V. Therefore, one must set up the circuit such that, in the dc mode, the maximum input value is scaled to be +3V.

A/D Converter Channel AC Mode Example

In another example, assume that a power circuit voltage is an ac quantity, and the range is as follows:

$$V_{i_max} = +/- 75 \text{ V}$$

The input mode of the A/D converter will be set to ac, and the input range is from -1.5V to +1.5V. Assume that the actual value of the voltage has a peak value of:

$$V_i = +/- 50 \text{ V}$$

Let the voltage sensor gain be 0.01. After the voltage sensor, the maximum value and the actual value of the input become:

$$V_{i_max_s} = +/- 0.75 \text{ V}$$

$$V_{i_s} = +/- 0.5 \text{ V}$$

Since the A/D converter input range is from -1.5V to +1.5V, this signal must be scaled before it is sent to the DSP. A conditioning circuit with a gain of 2 is needed (i.e. $1.5/0.75 = 2$). After the conditioning circuit and at the input of the DSP A/D converter, the maximum value and the actual value of the input become:

$$V_{i_max_c} = +/- 1.5 \text{ V}$$

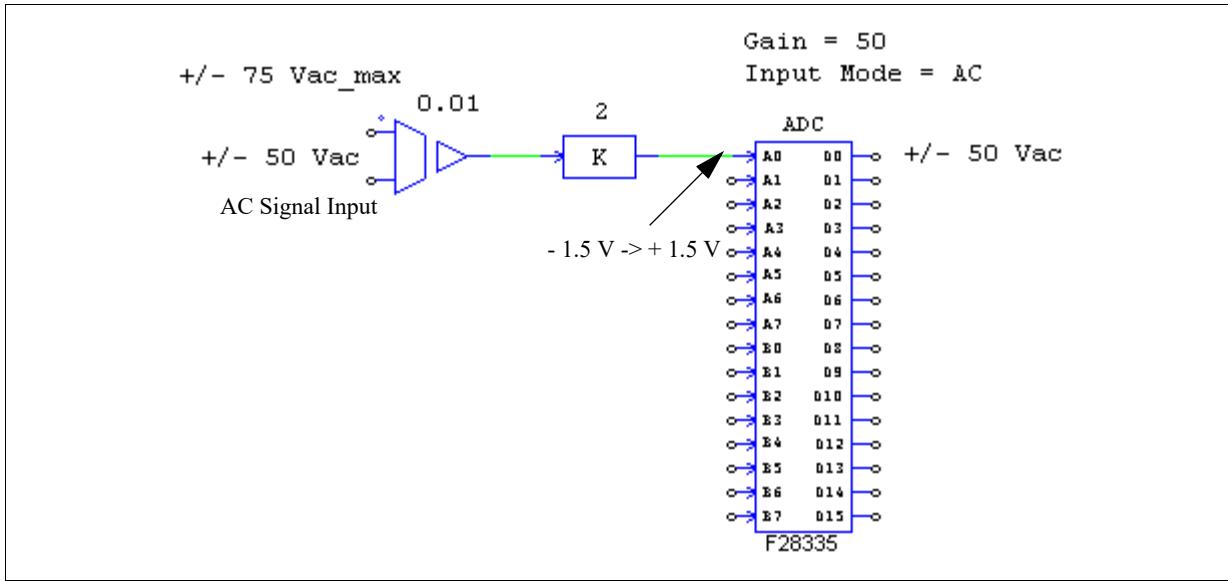
$$V_{i_s_c} = +/- 1 \text{ V}$$

The scaling block after the DSP A/D can be selected such that the original power circuit quantity is restored. In this example, a gain of 50 will be used. Note that this is the reciprocal of the combined gain of the voltage sensor and the conditioning circuit. At the A/D output, the maximum value and the actual value are:

$$V_{o_max} = +/- 75 \text{ V}$$

$$V_o = +/- 50 \text{ V}$$

The gain of the A/D channel in PSIM will be set to 50. The circuit connection and the settings are shown in the figure below.



Notice that in this circuit, the ac signal is sent to the A/D converter directly. This is because that, when the A/D input mode is set to ac, the input range is from -1.5V and +1.5V, and the function of the conditioning circuit that performs the dc offset is already included in the A/D converter block. In the actual hardware circuit, the ac signal will need to be scaled and offset so that the range is within 0V to +3V required by the DSP A/D converter.

Also, to ensure the correctness of the generated code for the hardware, the maximum peak value of the input must be scaled to 1.5V at the input port of the A/D converter.

6.9 Digital Input and Digital Output

F2833x has 88 general-purpose-input-output (GPIO) ports that can be configured as either digital inputs or digital output. In SimCoder, an 8-channel block is provided for digital input or output. Multiple 8-channel digital input/output blocks can be used in a schematic.

Image:



Attributes for Digital Input:

Parameters	Description
Port Position for Input i	Port position of the Input i , where i is from 0 to 7. It can be one of the 88 GPIO ports, from GPIO0 to GPIO87.
Use as External Interrupt	Indicate if this port is used as an external interrupt input.

Attributes for Digital Output:

Parameters	Description
Port Position for Output i	Port position of the Output i , where i is from 0 to 7. It can be one of the 88 GPIO ports, from GPIO0 to GPIO87.

Note that if a GPIO port is used as an input port, this same port cannot be used as another peripheral port. For example, if Port GPIO1 is assigned as digital input and it is also used as PWM1 output, an error will be reported.

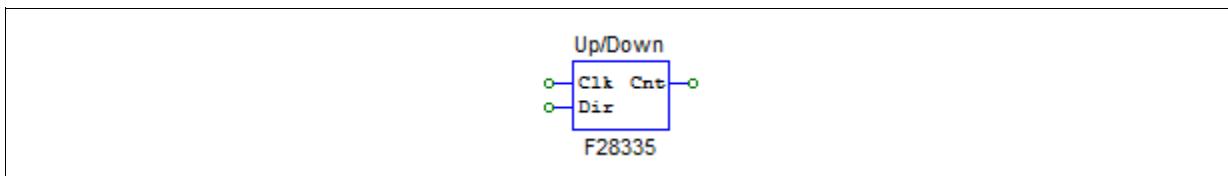
In the F2833x DSP, up to 7 external interrupt sources can be defined from ports GPIO0 to GPIO63 (specifically, up to 2 interrupt sources from Port GPIO0 to GPIO31, and up to 5 interrupt sources from Port GPIO32 to GPIO63). The priority of external interrupts in Port GPIO0 to GPIO31 is higher than the priority of the interrupt in Port GPIO32 to GPIO63.

6.10 Up/Down Counter

F2833x has 2 up/down counters. Counter 1 can be at either Port GPIO 20-21, or Port GPIO50-51. Counter 2 is at Port GPIO24-25.

Note that Counter 1 at Port GPIO20-21 and Port GPIO50-51 uses the same inner function blocks, and cannot be used at the same time.

Image:



Attributes:

Parameters	Description
Counter Source	Source of the counter. It can be one of the following: <ul style="list-style-type: none"> - <i>Counter 1 (GPIO20, 21)</i>: Counter 1 at Port GPIO20 and 21 is used. - <i>Counter 1 (GPIO50, 51)</i>: Counter 1 at Port GPIO50 and 51 is used. - <i>Counter 2 (GPIO24, 25)</i>: Counter 2 at Port GPIO24 and 25 is used.

In the image, "Clk" refers to the input clock signal, and "Dir" refers to the signal that defines the counting direction. When the "Dir" input is 1, the counter counts forward, and when the input is 0, the counter counts backward.

Note that the "Clk" input corresponds to the first port of the counter, and the "Dir" input corresponds to the second port. For example, for Counter 1 at Port GPIO20 and 21, GPIO20 is the "Clk" input and GPIO21 is the "Dir" input.

The output of the up/down counter gives the counter value.

Note that the up/down counter uses the same resource as the encoder, and the same GPIO ports cannot be used in a counter and encoder at the same time. For example, using both Encoder 1 and Up/Down Counter 1 will cause conflict and is not allowed.

6.11 Encoder and Encoder State

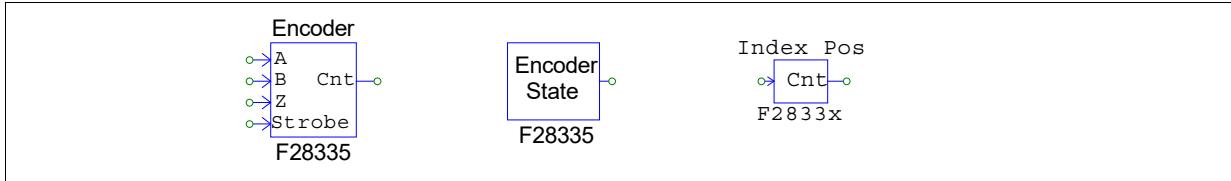
F2833x has 2 **Encoders**. Encoder 1 can be at either Port GPIO 20-21, or Port GPIO50-51. Encoder 2 is at Port GPIO24-25. Note that Encoder 1 at Port GPIO20-21 and at Port GPIO50-51 uses the same inner function blocks, and cannot be used at the same time. The output of the encoder gives the counter value.

The **Encoder State block** is used to indicate which input signal (either index signal or strobe signal) generates the interrupt. Also, hardware interrupt can be generated by the Z (index) signal and the strobe signal, and the output of the encoder state indicates which signal generates the interrupt. When the output is 0, the index signal

generates the interrupt. When the output is 1, the strobe signal generates the interrupt.

The **Encoder Index/Strobe Position block** is used to latch the encoder's initial position. When the input of this block is 0, the encoder counter is set to 0. The encoder will enable the latch when the input changes to 1. Encoder will latch the counter value when it meet the index/strobe event.

Image:



Attributes for Encoder:

Parameters	Description
Encoder Source	Source of the encoder. It can be one of the followings: - <i>Encoder 1 (GPIO20, 21)</i> : Encoder 1 at Port GPIO20 and 21 is used, with GPIO22 as Strobe and GPIO23 as Index (Z). - <i>Encoder 1 (GPIO50, 51)</i> : Encoder 1 at Port GPIO50 and 51 is used, with GPIO52 as Strobe and GPIO53 as Index (Z). - <i>Encoder 2 (GPIO24, 25)</i> : Encoder 2 at Port GPIO24 and 25 is used, with GPIO27 as Strobe and GPIO26 as Index (Z).
Use Z Signal	Define if the encoder uses the Z (or index) signal. It can be: - <i>No</i> : Not used. - <i>Yes (rising edge)</i> : The signal rising edge is used. - <i>Yes (falling edge)</i> : The signal falling edge is used.
Use Strobe Signal	Define if the encoder uses the strobe signal. - <i>No</i> : Not used. - <i>Yes (rising edge)</i> : The signal rising edge is used. - <i>Yes (rising/falling edge)</i> : Both the signal rising and falling edges are used.
Counting Direction	The counting direction can be either <i>Forward</i> or <i>Reverse</i> . When it is set to <i>Forward</i> , the encoder counts up. Otherwise, the encoder counts down.
Z Signal Polarity	Trigger polarity of Z signal. It can be: - <i>Active High</i> - <i>Active Low</i>
Strobe Signal Polarity	Trigger polarity of strobe signal. It can be: - <i>Active High</i> - <i>Active Low</i>
Encoder Resolution	Resolution of the external encoder hardware. If it is 0, the encoder counter will keep on counting and will not reset. If for example, the resolution is set to 4096, the counter will be reset to 0 after it reaches 4095.

Attributes for Encoder State:

Parameters	Description
Encoder Source	Define which encoder generates the interrupt. It can be one of the followings, must be the same Encoder used in the same schematic: - <i>Encoder 1 (GPIO20, 21)</i> : Encoder 1 at Port GPIO20 and 21 is used. - <i>Encoder 1 (GPIO50, 51)</i> : Encoder 1 at Port GPIO50 and 51 is used. - <i>Encoder 2 (GPIO24, 25)</i> : Encoder 2 at Port GPIO24 and 25 is used.

Attributes for Encoder Index/Strobe Pos:

Parameters	Description
Encoder Source	Source of the encoder. There are three sources, as listed below: - <i>Encoder 1 (GPIO20, 21)</i> : Encoder 1 at Port GPIO20 and 21 is used. - <i>Encoder 1 (GPIO50, 51)</i> : Encoder 1 at Port GPIO50 and 51 is used. - <i>Encoder 2 (GPIO24, 25)</i> : Encoder 2 at Port GPIO24 and 25 is used. Note that Encoder 1 (GPIO20, 21) and Encoder 1 (GPIO50, 51) use the same internal function blocks, and they cannot be used at the same time.
Latch Position	Latch counter type. It can be one of the following: - <i>IndexPos</i> : Z/index signal is used. - <i>StrobePos</i> : Strobe signal is used.
Position Type	Latch position. It can be one of the following: - <i>The first latched position</i> - <i>The current latched position</i>

6.12 Capture and Capture State

F2833x provides 6 captures. A capture can generate interrupt, and the interrupt trigger mode is defined by the interrupt block.

Image:



Attributes for Capture:

Parameters	Description
Capture Source	Source of the capture. There are in total 6 captures that use 14 designated GPIO ports, as listed below: - <i>Capture 1 (GPIO5, 24, 34)</i> - <i>Capture 2 (GPIO7, 25, 37)</i> - <i>Capture 3 (GPIO9, 26)</i> - <i>Capture 4 (GPIO11, 27)</i> - <i>Capture 5 (GPIO3, 48)</i> - <i>Capture 6 (GPIO1, 49)</i>
Event Filter Prescale	Event filter prescale. The input signal is divided by the selected prescale.
Timer Mode	Capture counter timer mode. It can be either <i>Absolute time</i> or <i>Time difference</i> .

Attributes for Capture State:

Parameters	Description
Capture Source	Source of the capture. It can be one of the 6 captures: <i>Capture 1</i> , <i>Capture 2</i> , ..., <i>Capture 6</i> .

The Capture State block output is either 1 or 0, where 1 means the rising edge and 0 means the falling edge.

6.13 Serial Communication Interface (SCI)

F2833x provides the function for serial communication interface (SCI). Through SCI, data inside the DSP can be transferred to a computer using an external RS-232 cable. PSIM provides all the necessary functions to transmit and receive data on both the DSP and computer sides, and to display the data on the computer. This provides a very convenient way to monitor, debug, and adjust the DSP code in real time.

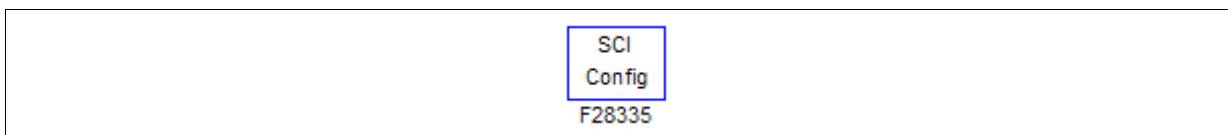
For more detailed descriptions on SCI and the monitoring function, please refer to the document "Tutorial - Using SCI for Real-Time Monitoring.pdf".

Three SCI function blocks are provided in SimCoder: *SCI Configuration*, *SCI Input*, and *SCI Output*, as described below.

6.13.1 SCI Configuration

The SCI Configuration block defines the SCI port, communication speed, parity check type, and data buffer size.

Image:



Attributes:

Parameters	Description
SCI Port	Define the SCI port. Different sets of GPIO ports that can be used for SCI, as listed below: - SCIA: GPIO28 and 35 in combination with GPIO29 and 36 - SCIB: GPIO9, 14, 18, and 22 in combination with GPIO11, 15, 19, and 23 - SCIC: GPIO62 and 63
Speed (bps)	SCI communication speed, in bps (bits per second). A list of preset speeds is provided at 200000, 115200, 57600, 38400, 19200, or 9600 bps. Or one can specify any other speed manually.
Parity Check	The parity check setting for error check in communication. It can be either <i>None</i> , <i>Odd</i> , or <i>Even</i> .
Output Buffer Size	Size of the data buffer allocated in DSP for SCI. The buffer is located in the RAM area, and each buffer cell stores one data point which consists of three 16-bit words (that is, 6 bytes, or 48 bits, per data point).

Note that the buffer size should be properly selected. On one hand, a large buffer is preferred in order to collect more data points so that more variables can be monitored over a longer period of time. On the other hand, the internal DSP memory is limited, and the buffer should not be too large to interfere with the normal DSP operation.

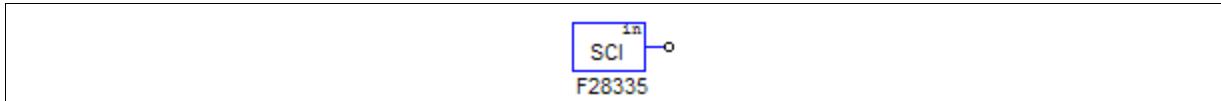
For more information on how to select the buffer size, please refer to the document "Tutorial - Using SCI for Waveform Monitoring.pdf".

6.13.2 SCI Input

The SCI Input block is used to define a variable in the DSP code that can be changed. The name of the SCI input variable will appear in the DSP Oscilloscope (under the **Utilities** menu), and the value can be changed at runtime via SCI.

The SCI input block provides a convenient way to change reference, or fine tune controller parameters, for example.

Image:



Attributes:

Parameters	Description
Initial Value	The initial value of the SCI input variable.

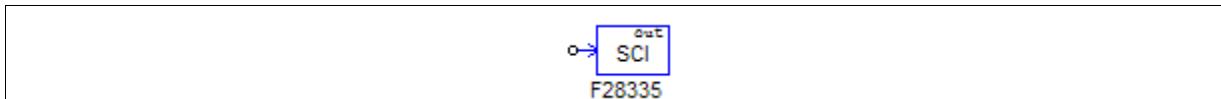
In a schematic, the SCI input behaves as a constant. While its value can be changed at runtime when the code is running on the DSP, the value will be fixed at the initial value in the simulation.

6.13.3 SCI Output

The SCI Output block is used to define a variable for display. When a SCI output block is connected to a node, the name of the SCI output block will appear in the DSP Oscilloscope (under the **Utilities** menu), and data of this variable can be transmitted from DSP to the computer via SCI at runtime, and the waveform can be displayed in the DSP Oscilloscope.

The SCI output block provides a convenient way to monitor DSP waveforms.

Image:



Attributes:

Parameters	Description
Data Point Step	It defines how frequent data is collected. If the Data Point Step is 1, every data point is collected and transmitted. If the Data Point Step is 10, for example, only one point of out every 10 points is collected and transmitted.

Note that if the Data Point Step is too small, there may be too many data points and it may not be possible to transmit them all. In this case, some data points will be discarded during the data transmission.

Also, the Data Point Step parameter is used only when the DSP Oscilloscope is in the *continuous* mode. When it is in the *snap-shot* mode, this parameter is ignored and every point is collected and transmitted.

In simulation, the SCI output behaves as a voltage probe.

6.14 Serial Peripheral Interface (SPI)

F2833x provides the functions for serial peripheral interface (SPI). By using the SPI blocks in the TI F2833x Target library, one can implement the function to communicate with external SPI devices (such as external A/D and D/A converters) easily and conveniently. Writing code manually for SPI devices is often a time-consuming and non-trivial task. With the capability to support SPI, PSIM greatly simplifies and speeds up the coding and hardware implementation process.

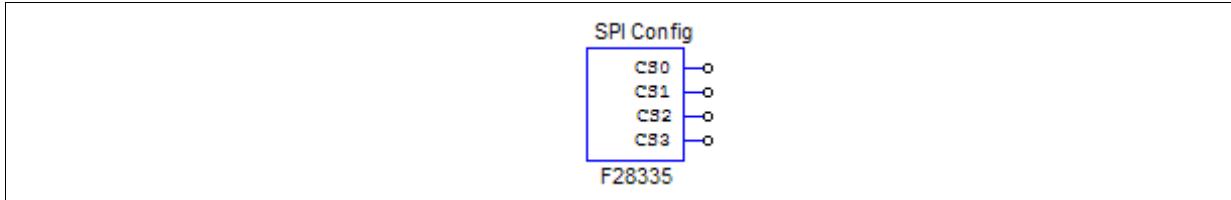
For more detailed descriptions on how to use SPI blocks, please refer to the document "*Tutorial - Using SCI for Real-Time Monitoring.pdf*".

Four SCI function blocks are provided in SimCoder: *SPI Configuration*, *SPI Device*, *SPI Input*, and *SPI Output*, as described below.

6.14.1 SPI Configuration

The SPI Configuration block defines the SPI port, the chip selection pins, and the SPI buffer size. It must be present in a schematic where SPI is used, and this block must be in the main schematic.

Image:



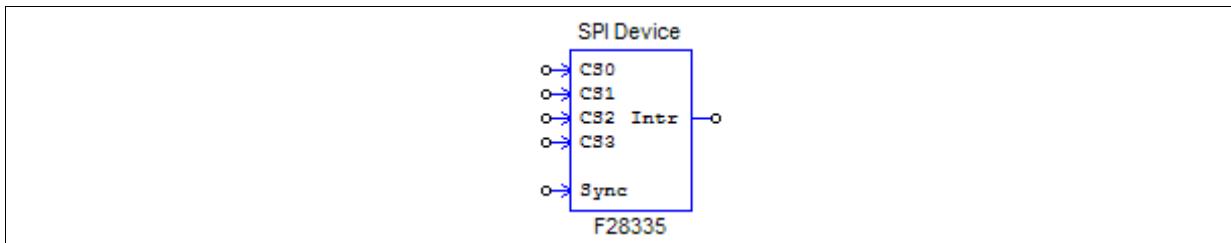
Attributes:

Parameters	Description
SPI Port	Define the SPI port. The SPI port can be either <i>GPIO16-19</i> or <i>GPIO54-57</i> .
Chip Select Pin0, 1, 2, and 3	The GPIO port of the chip select pin. PSIM supports up to 16 SPI devices, which requires four GPIO pins for chip select, as defined by Chip Select Pin0 to Pin3. These GPIO ports and the SPI slave transmit-enable pin SPISTE are used to generate the chip select signal.
SPI Buffer Size	The buffer size of the SPI commands. Each memory cell of the buffer saves the index of a SPI command. Normally, one can specify the buffer size as 1 plus the number of SPI commands (i.e. Start Conversion Command, Receiving Data Command, Sending Data Command, and Sync. Command) in all SPI Input/Output elements.

6.14.2 SPI Device

The SPI Device block defines the information of the corresponding SPI hardware device. The number of SPI Device blocks in the schematic must be the same as the number of SPI hardware devices.

Image:



Attributes:

Parameters	Description
Chip Select Pins	The state of the chip select pins corresponding to the SPI device. When the chip select pins are at this state, this SPI device is selected.
Communication Speed (MHz)	SPI communication speed, in MHz.

Clock Type	SPI clock type, as determined by the SPI hardware device. It can be one of the following: <ul style="list-style-type: none"> - <i>Rising edge without delay</i>: The clock is normally low, and data is latched at the clock rising edge. - <i>Rising edge with delay</i>: The clock is normally low, and data is latched at the clock rising edge with delay. - <i>Falling edge without delay</i>: The clock is normally high, and data is latched at the clock falling edge. - <i>Falling edge with delay</i>: The clock is normally high, and data is latched at the clock falling edge with delay.
Command Word Length	Word length, or the length of the significant bits, of SPI communication commands. It can be from 1 to 16 bits.
Sync. Active Mode	The triggering mode of the synchronization signal of the SPI device. It can be either <i>Rising edge</i> or <i>Falling edge</i> .
SPI Initial Command	The SPI command that initializes the SPI device.
Hardware Interrupt Mode	Specify the type of the interrupt signal that the SPI device generates. This is valid only when the SPI device's interrupt output node is connected to the input of a digital output element. It can be one of the following: <ul style="list-style-type: none"> - <i>No hardware interrupt</i> - <i>Rising edge</i> - <i>Falling edge</i>
Interrupt Timing	Specify how a SPI device generates interrupt when it completes conversion. It can be one of the following: <ul style="list-style-type: none"> - <i>No interrupt</i>: No interrupt is generated. In this case, DSP sends the command to a SPI input device. This device starts the conversion and returns the result in the same command - <i>Multiple interrupt in series</i>: Multiple interrupts are generated in series after each conversion. This is for a SPI device that has one A/D conversion unit and multiple input channels. In this case, DSP send the first conversion command, and the SPI device starts the conversion. When the conversion is complete, the SPI device will generate an interrupt. In the interrupt service routine, DSP will send a command to fetch the conversion result, and start a new conversion of another channel of the same SPI input device. - <i>One-time interrupt</i>: Only one interrupt is generated at the end of the conversion. This is for a SPI device that can perform multiple channel conversions in one request. In this case, DSP sends the command to the SPI input device, and the SPI device completes the conversion of multiple input channels. When all the conversions are complete, the SPI device will generate an interrupt.
Command Gaps (ns)	The gap between two SPI commands, in nsec.
Conversion Sequence	Define the names of the SPI input elements, separated by comma, that determine the conversion sequence. Note that this parameter is valid only when the SPI device generates multiple interrupts in series.

In a schematic, the chip select pins of all the SPI devices are connected to the chip select pins of the SPI Configuration block, without defining how the chip select logic is implemented. In the actual hardware, however, one would need to implement the corresponding chip select logic accordingly.

A SPI command consists of a series of 16-bit numbers separated by comma. In the 16-bit number, only the lower bits are the significant bits used by the command. For example, if the Command Word Length is 8, Bits 0 to 7 are the command, and Bits 8 to 15 are not used.

A SPI device can be either an input device or an output device. For example, an external A/D converter is an input device. Usually DSP will send one or multiple A/D conversion commands to the device, and then set the synchronization signal to start the conversion. The synchronization signal is reset at the next command of the

same device.

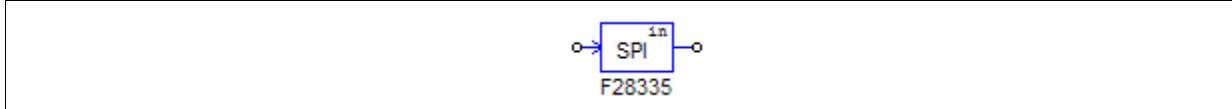
A SPI input device using the synchronization signal usually needs an interrupt pin to trigger DSP to enter the interrupt service routine.

On the other hand, an external D/A converter is an output device. Usually DSP sends one or multiple D/A conversion commands to the device, and then sets the synchronization signal to start the conversion. The synchronization signal is reset at the next command of the same device.

6.14.3 SPI Input

A SPI input device may have multiple input channels. The SPI Input block is used to define the properties of an input channel for SPI communication, and one SPI Input block corresponds to one input channel.

Image:



Attributes:

Parameters	Description
Device Name	Name of the SPI input device.
Start Conversion Command	Command to start conversion, in hex numbers, separated by comma (for example, 0x23,0x43,0x00).
Receiving Data Command	Command to receive data, in hex numbers, separated by comma (for example, 0x23,0x43,0x00).
Data Bit Position	Define where the data bits are in the receiving data string. The format is: <i>ElementName</i> = { <i>Xn</i> [MSB..LSB]} where - <i>ElementName</i> is the name of the SPI input device. If it is the current SPI input device, use <i>y</i> instead. - {} means that the item in the bracket repeats multiple times. - <i>Xn</i> is the <i>n_{th}</i> word received from the SPI input device, and <i>n</i> start from 0. - MSB..LSB defines the position of the significant bits in the word.
Input Range	Specify the parameter Vmax that defines the input range. This parameter is valid only when the SPI device is an A/D converter. If the device conversion mode is DC, the input ranges from 0 to Vmax. If the device conversion mode is AC, the input ranges from -Vmax/2 to Vmax/2.
Scale Factor	Output scale factor Kscale. If the scale factor is 0, the SPI device is not an A/D converter, and the result will be exactly the same as what DSP receives from SPI communication. Otherwise, the SPI device is an A/D converter, and the result is scaled based on this factor and the A/D conversion mode.
ADC Mode	The A/D conversion mode of the device. It can be either DC or AC. Note that this parameter is valid only when the device is an A/D converter.
Initial Value	The initial value of the input.

The formula for the *Data Bit Position* defines the data length of a SPI input device. For example, *y=x1[3..0]x2[7..0]*, means that the data length is 12, and the result is the lower 4 bits of the 2nd word and the lower 8 bits of the 3rd word. If the received data string is 0x12,0x78,0xAF, then the result is 0x8AF.

If the scale factor is not 0, the output will be scaled based on the following:

In the DC conversion mode:

- In simulation: $Output = Input \cdot K_{scale}$

- In hardware: $Output = \frac{Result \cdot V_{max} \cdot K_{scale}}{2^{Data_Length}}$

- In the AC conversion mode:

- In simulation: $Output = Input \cdot K_{scale}$

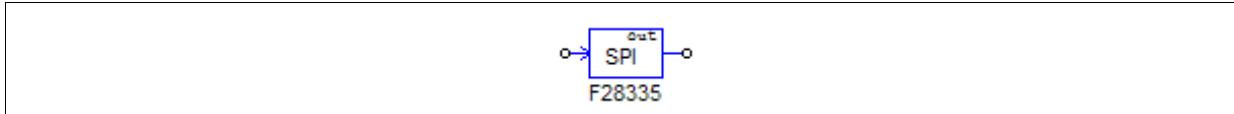
- In hardware: $Output = \frac{(Result - 2^{Data_Length-1}) \cdot V_{max} \cdot K_{scale}}{2^{Data_Length-1}}$

The parameter *Data_Length* is calculated from the Data Bit Position formula.

6.14.4 SPI Output

A SPI output device may have multiple output channels. The SPI Output block is used to define the properties of an output channel for SPI communication, and one SPI Output block corresponds to one output channel.

Image:



Attributes:

Parameters	Description
Device Name	Name of the SPI output device.
Scale Factor	Output scale factor K_{scale} . If the scale factor is 0, the SPI device is not a D/A converter, and the result will be exactly the same as what DSP receives from SPI communication. Otherwise, the SPI device is an D/A converter, and the result is scaled based on this factor and the D/A conversion mode.
Output Range	Specify the parameter V_{max} that defines the output range. This parameter is valid only when the SPI device is an D/A converter. If the device conversion mode is DC, the input ranges from 0 to V_{max} . If the device conversion mode is AC, the input ranges from $-V_{max}/2$ to $V_{max}/2$.
DAC Mode	The D/A conversion mode of the device. It can be either <i>DC</i> or <i>AC</i> . Note that this parameter is valid only when the device is a D/A converter.
Sending Data Command	Command to send the output data, in hex numbers, separated by comma (for example, 0x23,0x43,0x00).
Data Bit Position	Define where the data bits are in the sending data string. The format is: $ElementName = \{Xn[MSB..LSB]\}$ <p>where</p> <ul style="list-style-type: none"> - <i>ElementName</i> is the name of the SPI output device. If it is the current SPI output device, use <i>y</i> instead. - {} means that the item in the bracket repeats multiple times. - <i>Xn</i> is the n_{th} word sent to the SPI output device, and <i>n</i> start from 0. - <i>MSB..LSB</i> defines the position of the significant bits in the word.
Sync. Command	The command to synchronize output channels of the SPI output device, in hex numbers, separated by comma (for example, 0x23,0x43,0x00). This command is used when the SPI output device does not have the synchronization signal

The formula for the *Data Bit Position* defines the data length of a SPI output device. For example,

$y=x1[3..0]x2[7..0]$, means that the data length is 12, and the data is the lower 4 bits of the 2nd word and the lower 8 bits of the 3rd word. If the sending data string is 0x12,0x78,0xAF, then the data is 0x8AF.

If the scale factor is not 0, the output will be scaled based on the following:

In the DC conversion mode:

$$\text{- In simulation: } Output = Input \cdot K_{scale}$$

$$\text{- In hardware: } Output = \frac{Result \cdot K_{scale} \cdot 2^{Data_Length}}{V_{max}}$$

In the AC conversion mode:

$$\text{- In simulation: } Output = Input \cdot K_{scale}$$

$$\text{- In hardware: } Output = 2^{Data_Length} + \frac{Result \cdot K_{scale} \cdot 2^{Data_Length-1}}{V_{max}}$$

The parameter *Data_Length* is calculated from the Data Bit Position formula.

6.15 Controller Area Network (CAN) Bus

F2833x provides the function for CAN bus communication. PSIM provides the necessary functions to implement CAN bus.

Three function blocks are provided in SimCoder: *CAN Configuration*, *CAN Input*, and *CAN Output*, as described below.

6.15.1 CAN Configuration

The CAN Configuration block defines the CAN bus source, data byte order, and other settings.

Image:



Attributes:

Parameters	Description
CAN Source	The CAN source can be one of the two groups: CAN A and CAN B. CAN A uses a combination of GPIO 19 and 31 for transmit, versus GPIO 18 and 30 for receive. CAN B uses a combination of GPIO 8, 12, 16 and 20 for transmit, versus GPIO 10, 13, 17, and 21 for receive.
CAN Speed	Communication speed, in Hz. It can be set to one of the following preset values: 125kHz, 250kHz, 500kHz, and 1MHz Or you can set the speed manually by typing the text in the parameter field. Note that the speed should not exceed 1MHz.
Data Byte Order	The order of the data bytes. It can be one of the following: - Least significant byte 1st - Most significant byte 1st "Least significant byte 1st" means that the least significant byte is placed first; while "Most significant byte 1st" means that the most significant byte is placed first.

Number of Input Mailboxes	Number of input mailboxes
Checking Receive Mail Lost	If it is set to <i>Enable</i> , the error report function " <code>_ProcCanAErrReport(nErr)</code> " (for CAN A) will be called if the received mail is lost. This function will return the value of the error status nErr from the CAN register CANGIF0. If it is set to <i>Disable</i> , the error report function will not be called.
Checking Bus Off	If it is set to <i>Enable</i> , the error report function " <code>_ProcCanAErrReport(nErr)</code> " (for CAN A) will be called if the bus is in the off state. This function will return the value of the error status nErr from the CAN register CANGIF0. If it is set to <i>Disable</i> , the error report function will not be called.
Error Check Mode	The error check mode can be either <i>Passive</i> or <i>Active</i> . If it is in the error-passive mode, an interrupt will be generated when the error count reaches 128. If it is in the error-active mode, an interrupt will be generated every time.

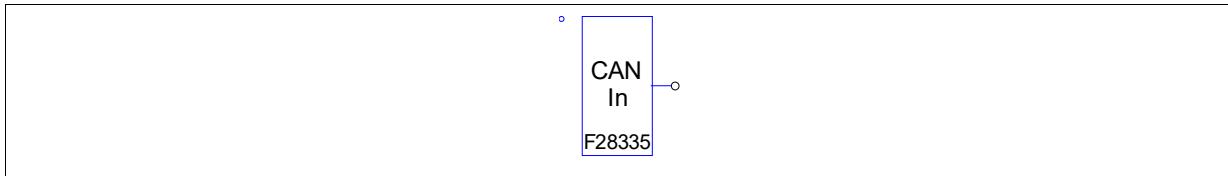
The returned status nErr in the function "`_ProcCanAErrReport(nErr)`" (for CAN A) is a 32-bit integer. It obtains its value from the Global Interrupt Flag Register CANGIF0. After returning from the function "`_ProcCanAErrReport(nErr)`", the register CANGIF0 will be cleared.

Also, if you wish to take actions on a specific error, you can add your own code within the "`_ProcCanAErrReport(nErr)`" function.

6.15.2 CAN Input

A CAN Input block receives CAN messages from a CAN bus.

Image:



Attributes:

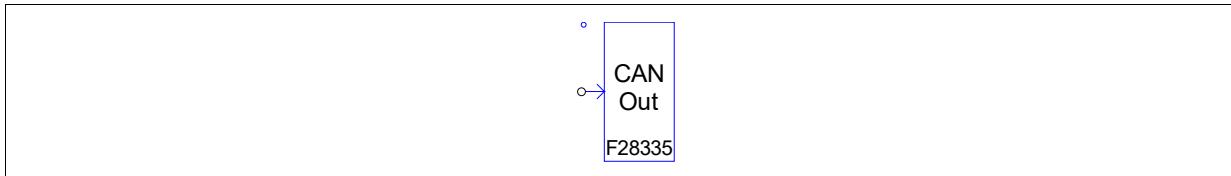
Parameters	Description
Number of Inputs	Number of CAN inputs. It can have up to 8 inputs.
CAN Source	The CAN source can be either CAN A or CAN B.
Use Extension ID	If this is set to <i>Yes</i> , the ID of a message is a 29-bit integer. If this is set to <i>No</i> , the ID of a message is a 11-bit integer.
Message ID	The ID of a message. It is an integer, for example, 0x23.
Local Mask	The mask for the message. It is an integer. If the bits of the message ID matches the bits of the mask, the message will be received. Otherwise, it will be ignored. For example, if the mask is 0x380 and the message ID is 0x389, this message will be received. But if the message ID is 0x480, the message will be ignored.
Overwrite Flag	It can be set to <i>Allow overwrite</i> or <i>Do not allow overwrite</i> . Assume a mailbox is configured to accept a message, and there is a new message coming from the CAN bus, while the old message is still in the mailbox and has not been processed yet. If the flag is set to "Allow overwrite", the new message will be accepted, and the old message will be overwritten. If the flag is set to "Do not allow overwrite", the new message will not be accepted, and the old one will be kept.

Receive Message Rate	The number of messages received by the input block in a specific period of time. For example, there are two input blocks, with the first input block receiving 20 messages per second, and the second input block receiving 30 messages per second. The parameter "Receive Message Rate" will be set to 20 for the first block, and 30 for the second block.
Input <i>i</i> Gain	The gain to the i_{th} input where i can be 1 to 8. The output is the input multiplied by the gain.
Input <i>i</i> Data Start Position	A message can have up to 8 data points. A data point can have 1 bit up to 32 bits. This defines the start position of the current data point in the message.
Input <i>i</i> Data End Position	This defines the end position of the current data point in the message.
Input <i>i</i> Data Type	The data type can be either <i>Float</i> , <i>Integer</i> , or <i>IQ1</i> to <i>IQ30</i> .
Input <i>i</i> Default Data	The initial value of the SCI input variable.

6.15.3 CAN Output

A CAN Output block transmits CAN messages to a CAN bus.

Image:



Attributes:

Parameters	Description
Number of Outputs	Number of CAN outputs. It can have up to 8 outputs.
CAN Source	The CAN source can be either CAN A or CAN B.
Use Extension ID	If this is set to <i>Yes</i> , the ID of a message is a 29-bit integer. If this is set to <i>No</i> , the ID of a message is a 11-bit integer.
Message ID	The ID of a message. It is an integer, for example, 0x23.
Trigger Type	The trigger type can be one of the following: <ul style="list-style-type: none"> - <i>No trigger</i>: No triggering - <i>Rising edge</i>: Triggering occurs at the rising edge of the trigger source. - <i>Falling edge</i>: Triggering occurs at the falling edge of the trigger source. - <i>Rising/falling edge</i>: Triggering occurs at both the rising edge and the falling edge of the trigger source. A rising/falling edge is considered to have occurred if the difference between the current value of the trigger source and the value at the last triggering instant is equal to or greater than 1.

Trigger Source	A trigger source can be either a constant or a global variable depending on the Trigger Type. If the Trigger Type is set to "No trigger", the trigger source defines the counter limit. For example, if the trigger source is a constant or a global value, and the value is 5, it means that triggering will occur once out of every 5 times. In another word, the data will be sent out once per every 5 cycles.
Output <i>i</i> Gain	If the Trigger Type is set to edge trigger, the trigger source can only be a global variable. Triggering will occur when the global variable has the rising or falling edge or both depending on the Trigger Type. The gain to the i_{th} input where i can be 1 to 8. The output is the input multiplied by the gain.
Output <i>i</i> Data Start Position	A message can have up to 8 data points. A data point can have 1 bit up to 32 bits. This defines the start position of the current data point in the message.
Output <i>i</i> Data End Position	This defines the end position of the current data point in the message.
Output <i>i</i> Data Type	The data type can be either <i>Float</i> , <i>Integer</i> , or <i>IQ1</i> to <i>IQ30</i> .
Output <i>i</i> Default Data	The initial value of the SCI output variable.

6.16 Interrupt Time

The interrupt time block is used to measure the time interval of an interrupt service routine.

Attributes:

Parameters	Description
Time Output Method	Define how interrupt time is measured. It can be one of the following: <ul style="list-style-type: none"> - <i>SCI (time used)</i>: Using SCI. Time used by the interrupt service routine, in DSP clock count, is measured and is sent out via SCI output. - <i>SCI (time remaining)</i>: Using SCI. Time remaining in the interrupt service routine, in DSP clock count, is measured and is send out via SCI output. The time remaining is defined as the time from the end of the current interrupt to the beginning of the next interrupt. - <i>GPIO0 to GPIO87</i>: Using a GPIO port. A pulse is generated at the specified GPIO port. The pulse is set to high when entering the interrupt, and set to low when exiting the interrupt. An oscilloscope can be used to measure the width of the pulse.
Sampling Frequency	Sampling frequency of the interrupt service routine, in Hz.

When SCI is used, the value is the count of the DSP clock. For example, if the value is 7500, for a 150-MHz DSP clock, the interrupt time will be: $7500 / 150M = 50 \text{ us}$.

6.17 Project Settings and Memory Allocation

When generating the code for the F2833x Hardware Target, SimCoder also creates the complete project files for the TI Code Composer Studio (CCS) development environment where the code will be compiled, linked, and uploaded to the DSP.

At the present, CCS version 3.3 is supported. Assuming that the PSIM schematic file is "test.sch", after the code generation, a sub-folder called "test (C code)" will be generated in the directory of the schematic file, and sub-folder will contain the following files:

- test.c Generated C code
- PS_bios.h: Header file for the SimCoder F2833x library
- passwords.asm: File for specifying the DSP code password

- test.pjt: Project file for Code Composer Studio
- F2833x_Headers_nonBIOS.cmd: Peripheral register linker command file
- F28335_FLASH_Lnk.cmd: Flash memory linker command file
- F28335_FLASH_RAM_Lnk.cmd: Flash RAM memory linker command file
- F28335_RAM_Lnk.cmd: RAM memory linker command file

Note: The names of the link command files are assigned with the target hardware if it is not F28335. For example, if the target hardware is F28334, the file names will be *F28334 FLASH Lnk.cmd*, *F28334 FLASH RAM Lnk.cmd*, and *F28334RAM Lnk.cmd* accordingly.

Besides, the project also needs the following library files:

- PS_bios.lib: SimCoder F2833x library, located in the PSIM folder
- C28x_FPU_FastRTS_beta1.lib: TI fast floating-point library, located in the PSIM \lib sub-folder

These library files will be copied automatically to the project folder when the code is generated.

Each time code generation is performed, the .c file and .pjt file (in this example, "test.c" and "test.pjt") will be created. If you have made changes manually to these two files, be sure to copy the changed files to a different location. Otherwise the changes will be overwritten when code generation is performed next time.

Project Setting:

In the Code Composer Studio project file, the following settings are provided:

- *RAM Debug*: To compile the code in the debug mode and run it in the RAM memory
- *RAM Release*: To compile the code in the release mode and run it in the RAM memory
- *Flash Release*: To compile the code in the release mode and run it in the flash memory
- *Flash RAM Release*: To compile the code in the release mode and run it in the RAM memory

When RAM Debug or RAM Release is selected, CCS uses the linker command file F28335_RAM_Lnk.cmd to allocate the program and data space.

When Flash Release is selected, CCS uses the linker command file F28335_FLASH_Lnk.cmd to allocate the program and data space.

When Flash RAM Release is selected, CCS uses the linker command file F28335_FLASH_RAM_Lnk.cmd to allocate the program and data space. The memory allocation is the same as in RAM Release.

The code compiled in the release mode is faster than the code in the debug mode. Also, the code in RAM Release or Flash RAM Release is the fastest. The code in RAM Debug is slower, and the code in Flash Release is the slowest. In a development, normally one would start with RAM Debug for easy debugging. Then switch to RAM Release and consequently to Flash Release or Flash RAM Release.

Memory Allocation:

In the generated link files, the memory allocation is defined in the following way.

With the RAM Debug, RAM Release, and Flash RAM Release settings:

RAM Memory

0x0000 - 0x07FF (2K)
interrupt vectors
stack
0x8000 - 0xFFFF (32K*)
program and data space

With the Flash Release setting:

RAM Memory	Flash Memory
0x0000 - 0x07FF (2K)	0x300000 - 0x33FFFF (256K**)
interrupt vectors	program
stack	password
0x8000 - 0xFFFF (32K*)	etc.
data space	

Notes:

- * The RAM memory predefined by SimCoder for program and data space is:
 - For F28335, F28334: from 0x8000 to 0xFFFF (32K)
 - For F28332, from 0x8000 to 0xDFFF (16K)
 - If the combined program and data space exceeds the size of the RAM space, Flash Release must be selected as the project setting.
- ** The flash memory predefined by SimCoder for program space is:
 - For F28335: from 0x300000 to 0x33FFFF (256K)
 - For F28334: from 0x320000 to 0x33FFFF (128K)
 - For F28332: from 0x330000 to 0x33FFFF (64K)

F2803x Hardware Target

7.1 Overview

With the F2803x Hardware Target, SimCoder can generate code that is ready to run on any hardware boards based on Texas Instruments' F2803x fixed-point DSP.

The F2803x Hardware Target will work with all F2803x packages.

The F2803x Hardware Target library includes the following function blocks:

- PWM generators: 3-phase, 2-phase, 1-phase, and APWM
- Variable frequency PWM
- Start/Stop functions for PWM generators
- Trip-one and trip-zone state
- A/D converter
- Comparator input, output, and DAC
- Digital input and output
- Up/Down counter
- Encoder and encoder state
- Capture and capture state
- SCI configuration, Input, and output
- SPI configuration, device, input, and output
- CAN configuration, input, and output
- Interrupt Time
- DSP clock
- Hardware configuration

When generating the code for a system that has multiple sampling rates, SimCoder will use the interrupts of the PWM generators for the PWM sampling rates. For other sampling rates in the control system, it will use the Timer 1 interrupt first, and then Timer 2 interrupt if needed. If there are more than three sampling rates in the control system, the corresponding interrupt routines will be handled in the main program by software.

In TI F2803x, PWM generators can generate hardware interrupt. SimCoder will search and group all the elements that are connected to the PWM generator and have the same sampling rate as the PWM generator. These elements will be automatically placed and implemented in an interrupt service routine in the generated code.

In addition, digital input, encoder, capture, and trip-zone can also generate hardware interrupt. Each hardware interrupt must be associated with an interrupt block (described in Section 4.5 of this Manual), and each interrupt block must be associated with an interrupt service routine (a subcircuit that represents the interrupt service routine). For example, if a PWM generator and a digital input both generate interrupt, there should be one interrupt block and one interrupt service routine for each of them.

The definitions of the elements in the F2803x Hardware Target library are described in this Chapter.

The figure below shows the F2803x 80-pin PN QFP port assignment.

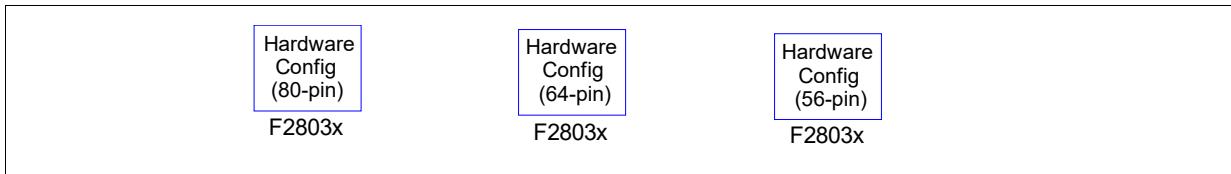
F2803x 80-Pin PN QFP Port Assignment

GPIO22/EQEP1S/LINTXA	1	40	GPIO28/SCIRXDA/SDAA/TZ2
GPIO32/SDAA/EPWMSYNCI/ADCSOCAO	2	39	GPIO9/EPWM5B/LINTXA/HRCAP1
GPIO33/SCLA/EPWMSYNCO/ADCSOCBO	3	38	TEST2
GPIO23/EQEP1I/LINRXA	4	37	GPIO26/HRCAP1/SPICLKB
GPIO42/COMP1OUT	5	36	VDDIO
GPIO43/COMP2OUT	6	35	VSS
VDD	7	34	GPIO29/SCITXDA/SCLA/TZ3
Vss	8	33	GPIO30/CANRXA
XRS	9	32	GPIO31/CANTXA
TRST	10	31	GPIO27/HRCAP2/SPISTEB
ADCINA7	11	30	ADCINB7
ADCINA6/COMP3A/AIO6	12	29	ADCINB6/COMP3B/AIO14
ADCINA5	13	28	ADCINB5
ADCINA4/COMP2A/AIO4	14	27	ADCINB4/COMP2B/AIO12
ADCINA3	15	26	ADCINB3
ADCINA2/COMP1A/AIO2	16	25	ADCINB2/COMP1B/AIO10
ADCINA1	17	24	ADCINB1
ADCINA0	18	23	ADCINB0
VREFHI	19	22	VREFLO
VDDA	20	21	VSSA
GPIO11/EPWM6B/LINRXA/HRCAP2	61	60	GPIO36/TMS
GPIO5/EPWM3B/SPISIMO/A/ECAP1	62	59	GPIO35/TDI
GPIO4/EPWM3A	63	58	GPIO37/TDO
GPIO40/EPWM7A	64	57	GPIO38/TCK/XCLKIN
GPIO10/EPWM6A/ADCSOCBO	65	56	GPIO39
GPIO3/EPWM2B/SPISOMIA/COMP2OUT	66	55	GPIO19/XCLKIN/SPISTEA/LINRXA/ECAP1
GPIO2/EPWM2A	67	54	VDD
GPIO1/EPWM1B/COMP1OUT	68	53	VSS
GPIO0/EPWM1A	69	52	X1
VDDIO	70	51	X2
VSS	71	50	GPIO6/EPWM4A/EPWMSYNCI/EPWMSYNCO
VDD	72	49	GPIO7/EPWM4B/SCIRXDA
VREGENZ	73	48	GPIO41/EPWM7B
GPIO34/COMP2OUT/COMP3OUT	74	47	GPIO12/TZ1/SCITXDA/SPISIMOB
GPIO15/TZ1/LINRXA/SPISTEB	75	46	GPIO16/SPISIMO/A/TZ2
GPIO13/TZ2/SPISOMIB	76	45	GPIO44
GPIO14/TZ3/LINTXA/SPICLKB	77	44	GPIO25/SPISOMIB
GPIO20/EQEP1A/COMP1OUT	78	43	GPIO8/EPWM5A/ADCSOCAO
GPIO21/EQEP1B/COMP2OUT	79	42	GPIO17/SPISOMIA/TZ3
GPIO24/ECAP1/SPISIMOB	80	41	GPIO18/SPICLKA/LINTXA/XCLKOUT

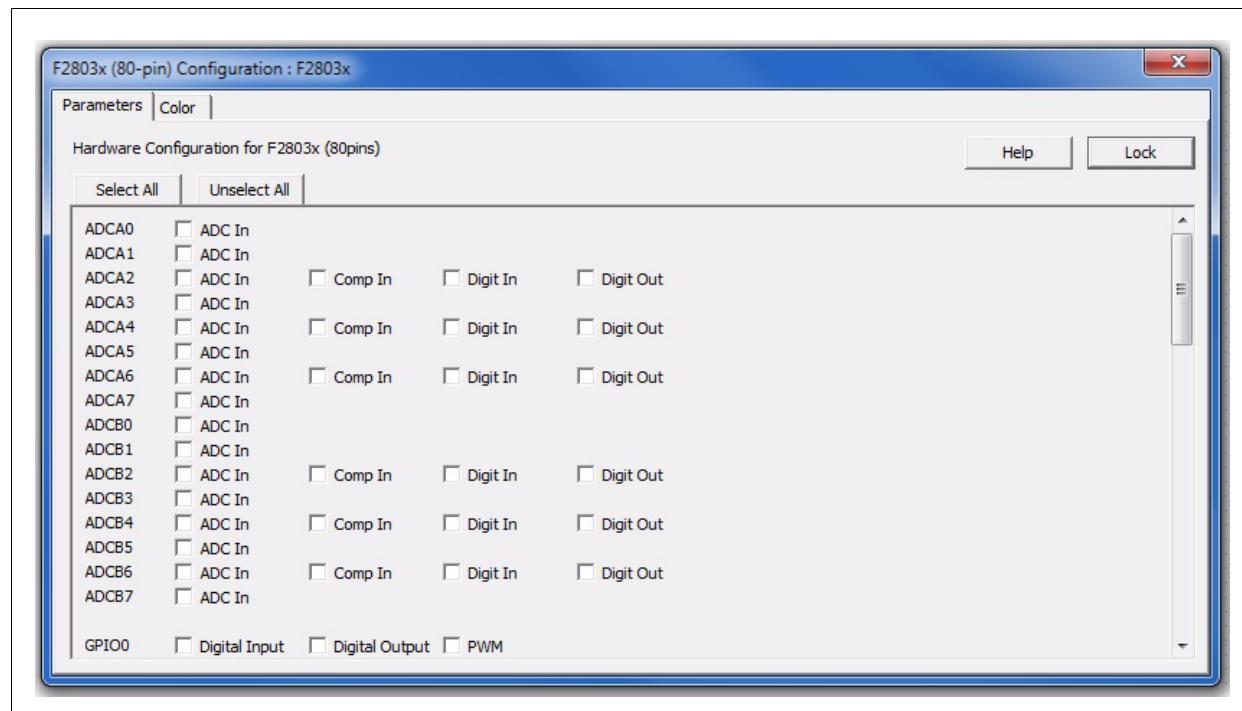
7.2 Hardware Configuration

F2803x provides a 16 channel analog inputs and up to 45 individually programmable multiplexed GPIO ports. Many of the GPIO ports can perform one of several functions. For example, port GPIO1 can be used either as a digital input, a digital output, or a PWM output. Some of the 16 A/D channels can also be defined as either digital input, digital output, or comparator input. Therefore, user must assign the functions to the GPIO ports correctly according to the PSIM circuit schematic.

Image:



The dialog window of the block is shown below:

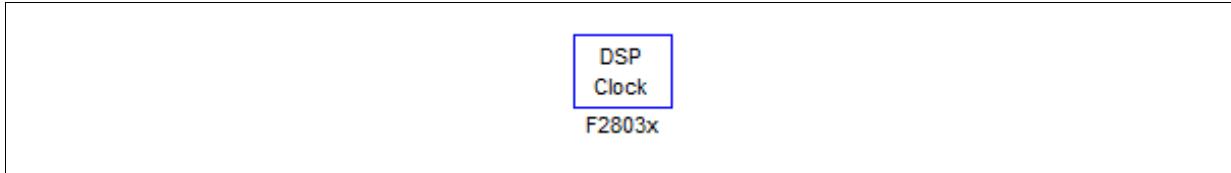


The Hardware Configuration block is for user to specify the I/O ports of the F2803x hardware. Every port to be used must be assigned correctly. The ports not in use can be left unchecked.

For each GPIO port, a check box is provided for each of its available function. When a box is checked, the GPIO port is configured for that particular function. For example, if the checkbox for "Digital Input" is checked for port GPIO1, this port is configured as a digital input, and hence, cannot be used for any other functions. If it is used as a PWM output in the PSIM circuit schematic, an error message will be generated.

7.3 DSP Clock

The DSP Configuration block defines the external clock frequency and the speed of the F2803x DSP, as well as the program space size.

Image:**Attributes:**

Parameters	Description
DSP Clock Source	There are five ways of providing system clock to F2803x. They are as follows: - Internal oscillator 1 - Internal oscillator 2 - External oscillator - External clock (GPIO19) - External clock (GPIO38)
External Clock (MHz)	Frequency of the external clock on the DSP board, in MHz. The frequency must be an integer, and the maximum frequency allowed is 10 MHz. This parameter is ignored if the DSP clock source is selected to be internal oscillator 1 or 2.
DSP Speed (MHz)	DSP Speed, in MHz. The speed must be an integer, and must be an integer multiple of the external clock frequency, from 1 to 12 times. The maximum DSP speed allowed is 60 MHz.

If a DSP Configuration block is not used in a circuit, the default values of the DSP Configuration block are used.

7.4 PWM Generators

F2803x contains 7 sets of PWM modules, and each set of PWM module has two output ports:

- PWM 1 (GPIO 0 and 1)
- PWM 2 (GPIO 2 and 3)
- PWM 3 (GPIO 4 and 5)
- PWM 4 (GPIO 6 and 7)
- PWM 5 (GPIO 8 and 9)
- PWM 6 (GPIO 10 and 11)
- PWM 7 (GPIO 40 and 41)

The two outputs of each PWM module usually are complementary to each other. For example PWM 1 has a positive output PWM 1A and a negative output PWM 1B, except when the PWM module is set in one of a few special operation modes.

In SimCoder, these 7 PWM's can be used in the following ways:

- Two 3-phase PWM generators: PWM 123 (consisting of PWM 1, 2, and 3) and PWM 456 (consisting of PWM 4, 5, and 6);
- Seven 2-phase PWM generators: PWM 1, 2, 3, 4, 5, 6, and 7, with the two outputs of each PWM generator not in a complementary way, but in special operation mode.
- 1-phase PWM generators: PWM 1, 2, 3, 4, 5, 6, and 7, with two outputs complementary to each other.
- 1-phase PWM generators with phase shift: PWM 2, 3, 4, 5, and 6, with two outputs complementary to each other.

These PWM generators can trigger the A/D converter, and use trip-zone signals.

Beside the PWM generators described above, there are also 6 APWM generators that use the same resources as the captures. These PWM generators have restricted functionality as compared to the 6 PWM generators (PWM

1 to 6) as they can not trigger the A/D converter and can not use trip-zone signals. Also, because of the common resources, when a particular port is used for the capture, it can not be used for the PWM generator.

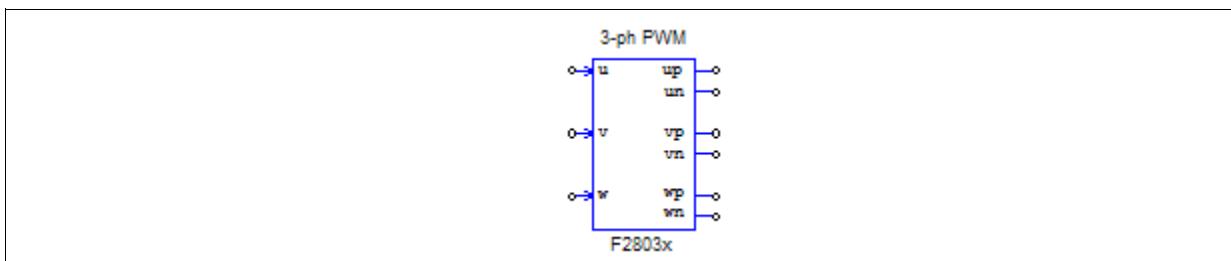
Note that all the PWM generators in SimCoder include one switching period delay internally. That is, the input value of a PWM generator is delayed by one cycle before it is used to update the PWM output. This delay is needed to simulate the delay inherent in the DSP hardware implementation.

PWM generators have a parameter called "PWM Freq. Scaling Factor". It can be set to 1 to 100. The hardware limit is 3. If the scaling factor is greater than 3, PWM will use an unused PWM to generator interrupt at the sampling frequency. This unused PWM is only used to generate a periodic interrupt, and its outputs can still be used for other functions. If there is no unused PWM in the system, a timer will be used.

7.4.1 3-Phase PWM

In the 3-phase PWM generator image, "u", "v", and "w" refer to the three phases (alternatively they are called Phase "a", "b", and "c"). The letter "p" refers to the positive output, and "n" refers to the negative output. For example, for 3-phase PWM 123, "up" is PWM1A, and "un" is PWM1B.

Image:



Attributes:

Parameters	Description
PWM Source	Source of the PWM generator. It can be either "3-phase PWM 123" that uses PWM 1 to 3, or "3-phase PWM 456" that uses PWM 4 to 6.
Dead Time	The dead time T_d for the PWM generator, in sec.
Sampling Frequency	Sampling frequency of the PWM generator, in Hz. The calculation is done and the PWM signal duty cycle is updated based on this frequency.
PWM Freq. Scaling Factor	The ratio between the PWM switching frequency and the sampling frequency. It can be from 1 to 100. For example, if the sampling frequency is 50 kHz and the scaling factor is 3, the PWM switching frequency will be 150 kHz. That is switches will operate at 150 kHz. But gating signals will be updated at 50 kHz, or once per 3 switching cycles.
Carrier Wave Type	The carrier wave type and the initial PWM output state. It can be one of the following: <ul style="list-style-type: none"> - <i>Triangular (start low)</i>: Triangular wave, and the initial PWM output state is low. - <i>Triangular (start high)</i>: Triangular wave, and the initial output state is high. - <i>Sawtooth (start low)</i>: Sawtooth wave, and the initial output state is low. - <i>Sawtooth (start high)</i>: Sawtooth wave, with the initial output state is high.
Trigger ADC	Setting whether for the PWM generator to trigger the A/D converter. It can be one of the following: <ul style="list-style-type: none"> - <i>Do not trigger ADC</i>: PWM does not trigger the A/D converter. - <i>Trigger ADC</i>: PWM will trigger A/D converter.

ADC Trigger Position Use Trip-Zone <i>i</i>	<p>The A/D trigger position ranges from 0 to a value less than 1. When it is 0, the A/D converter is triggered at the beginning of the PWM cycle, and when it is 0.5, the A/D converter is triggered at the 180° position of the PWM cycle.</p>
PWMA DC Trip Src1(DCAH)	<p>Define whether the PWM generator uses the i_{th} trip-zone signal or not, where i ranges from 1 to 6. It can be one of the following:</p> <ul style="list-style-type: none"> - <i>Disable Trip-Zone <i>i</i></i>: Disable the i_{th} trip-zone signal. - <i>One shot</i>: The PWM generator uses the trip-zone signal in the one-shot mode. Once triggered, the PWM must be started manually. - <i>Cycle by cycle</i>: The PWM generator uses the trip-zone signal in the cycle-by-cycle basis. The trip-zone signal is effective within the current cycle, and PWM will automatically re-start in the next cycle.
PWMA DC Trip Src1(DCAL)	<p>Digital compare (DC) trip source DCAH for PWMA. For a 3-phase PWM generator, PWMA refers to the outputs "up", "vp", and "wp" for the 3 top switches. The PWM channel may have up to two DC trip sources: DCAH and DCAL. The trip source can be one of the following:</p> <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>Trip-zone 1</i>: PWM uses trip-zone 1 as digital compare input signal. - <i>Trip-zone 2</i>: PWM uses trip-zone 2 as digital compare input signal. - <i>Trip-zone 3</i>: PWM uses trip-zone 3 as digital compare input signal. - <i>Comparator 1</i>: PWM uses Comparator 1 as digital compare input signal. - <i>Comparator 2</i>: PWM uses Comparator 2 as digital compare input signal. - <i>Comparator 3</i>: PWM uses Comparator 3 as digital compare input signal.
PWMA 1-shot Evt (DCAEVT1)	<p>Digital compare (DC) trip source DCAL for PWMA. The trip source can be one of the following:</p> <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>Trip-zone 1</i>: PWM uses trip-zone 1 as digital compare input signal. - <i>Trip-zone 2</i>: PWM uses trip-zone 2 as digital compare input signal. - <i>Trip-zone 3</i>: PWM uses trip-zone 3 as digital compare input signal. - <i>Comparator 1</i>: PWM uses Comparator 1 as digital compare input signal. - <i>Comparator 2</i>: PWM uses Comparator 2 as digital compare input signal. - <i>Comparator 3</i>: PWM uses Comparator 3 as digital compare input signal.
PWMA CBC Evt (DCAEVT1)	<p>Define how the one-shot signal is used for the DC trip signal of PWMA. The active level of the DC trip signal can be selected from the following:</p> <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>Trip source 1 is low</i>: PWM is tripped if the source signal 1 is low. - <i>Trip source 1 is high</i>: PWM is tripped if the source signal 1 is high. - <i>Trip source 2 is low</i>: PWM is tripped if the source signal 2 is low. - <i>Trip source 3 is high</i>: PWM is tripped if the source signal 2 is high. - <i>Trip source 1 low & source 2 high</i>: PWM is tripped if the source 1 signal is low and at the same time source 2 signal is high.
	<p>Define how the cycle-by-cycle signal is used for the DC trip signal of PWMA. The active level of the DC trip signal can be selected from the following:</p> <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>Trip source 1 is low</i>: PWM is tripped if the source signal 1 is low. - <i>Trip source 1 is high</i>: PWM is tripped if the source signal 1 is high. - <i>Trip source 2 is low</i>: PWM is tripped if the source signal 2 is low. - <i>Trip source 3 is high</i>: PWM is tripped if the source signal 2 is high. - <i>Trip source 1 low & source 2 high</i>: PWM is tripped if the source 1 signal is low and at the same time source 2 signal is high.

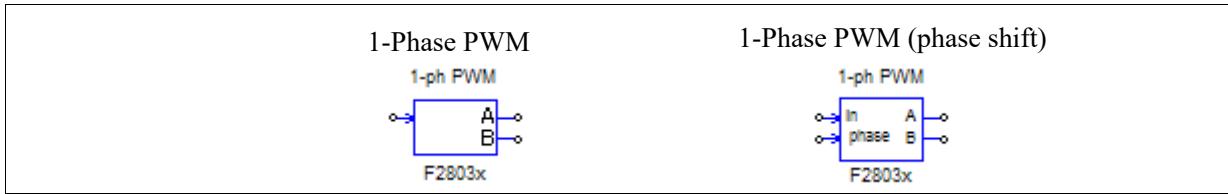
PWMB DC Trip Src1(DCBH)	Digital compare (DC) trip source DCBH for PWMB. For a 3-phase PWM generator, PWMB refers to the outputs "un", "vn", and "wn" for the 3 bottom switches. The PWM channel may have up to two DC trip sources: DCBH and DCBL. The trip source can be one of the following: <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>Trip-zone 1</i>: PWM uses trip-zone 1 as digital compare input signal. - <i>Trip-zone 2</i>: PWM uses trip-zone 2 as digital compare input signal. - <i>Trip-zone 3</i>: PWM uses trip-zone 3 as digital compare input signal. - <i>Comparator 1</i>: PWM uses Comparator 1 as digital compare input signal. - <i>Comparator 2</i>: PWM uses Comparator 2 as digital compare input signal. - <i>Comparator 3</i>: PWM uses Comparator 3 as digital compare input signal.
PWMB DC Trip Src1(DCBL)	Digital compare (DC) trip source DCBL for PWMB. The trip source can be one of the following: <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>Trip-zone 1</i>: PWM uses trip-zone 1 as digital compare input signal. - <i>Trip-zone 2</i>: PWM uses trip-zone 2 as digital compare input signal. - <i>Trip-zone 3</i>: PWM uses trip-zone 3 as digital compare input signal. - <i>Comparator 1</i>: PWM uses Comparator 1 as digital compare input signal. - <i>Comparator 2</i>: PWM uses Comparator 2 as digital compare input signal. - <i>Comparator 3</i>: PWM uses Comparator 3 as digital compare input signal.
PWMB 1-shot Evt (DCBEVT1)	Define how the one-shot signal is used for the DC trip signal of PWMB. The active level of the DC trip signal can be selected from the following: <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>Trip source 1 is low</i>: PWM is tripped if the source signal 1 is low. - <i>Trip source 1 is high</i>: PWM is tripped if the source signal 1 is high. - <i>Trip source 2 is low</i>: PWM is tripped if the source signal 2 is low. - <i>Trip source 3 is high</i>: PWM is tripped if the source signal 2 is high. - <i>Trip source 1 low & source 2 high</i>: PWM is tripped if the source 1 signal is low and at the same time source 2 signal is high.
PWMB CBC Evt (DCBEVT1)	Define how the cycle-by-cycle signal is used for the DC trip signal of PWMB. The active level of the DC trip signal can be selected from the following: <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>Trip source 1 is low</i>: PWM is tripped if the source signal 1 is low. - <i>Trip source 1 is high</i>: PWM is tripped if the source signal 1 is high. - <i>Trip source 2 is low</i>: PWM is tripped if the source signal 2 is low. - <i>Trip source 3 is high</i>: PWM is tripped if the source signal 2 is high. - <i>Trip source 1 low & source 2 high</i>: PWM is tripped if the source 1 signal is low and at the same time source 2 signal is high.
DC Event Filter Source	Source of the digital compare event filter. It can be one of the following: <ul style="list-style-type: none"> - <i>Do not use</i>: Do not use DC event filter. - <i>DCAEVT1</i>: Use DCAEVT1 as the filter source. - <i>DCAEVT2</i>: Use DCAEVT2 as the filter source. - <i>DCBEVT1</i>: Use DCBEVT1 as the filter source. - <i>DCBEVT2</i>: Use DCBEVT2 as the filter source. - <i>DCAEVT2</i>: PWM is tripped if the source signal 1 is high.
Blanking Window Pos (us)	Blanking window start position in a PWM period, in us.
Blanking Window Width (us)	Width of the blanking window, in us. The width is limited by the hardware, and can be calculated as below: $\text{255} / \text{CPU Frequency}$ <p>For example, when the CPU speed is 90MHz, the width range is 0 to 2.83us.</p>

Blanking Window Range	Specify how the blanking window is applied. It can be one of the following: - <i>In the window</i> : The blanking action is applied with the window defined (from the start position for a window width defined) - <i>Out of window</i> : The blanking action is applied outside the window defined.
Applying Event Filtering	Specify how event filtering is applied to digital compare events. The event filtering can be applied to any combinations of the following digital compare events: DCAEVT1, DCAEVT2, DCBEVT1, and DCBEVT2
Trip Action	Define how the PWM generator responds to the trip action. It can be one of the following: - <i>High impedance</i> : PWM outputs in high impedance - <i>PWM A high & B low</i> : Set PWM A high and B low. - <i>PWM A low & B high</i> : Set PWM A low and B high. - <i>No action</i> : No action taken.
Peak-to-Peak Value	Peak-to-peak value V_{pp} of the carrier wave
Offset Value	DC offset value V_{offset} of the carrier wave
Initial Input Value u, v, w	Initial value of 3-phase inputs u , v , and w
Start PWM at Beginning	When it is set to "Start", PWM will start right from the beginning. If it is set to "Do not start", one needs to start PWM using the "Start PWM" function.
Simulation Output Mode	The simulation output mode can be set to <i>Switching mode</i> or <i>Average mode</i> . When it is set to "Switching mode", the outputs of the PWM block are PWM signals. When it is set to "Average mode", the outputs of the PWM block are average mode signals. In the average mode, if the carrier wave is from negative to positive, and the absolute values of the negative peak and the positive peak are equal (for example, the carrier wave is from -1 to +1, or from -5 to +5), the modulation is considered as an ac signal modulation. Otherwise, the modulation is considered as a dc signal modulation. For example, modulation in a 3-phase or single-phase inverter is an ac modulation, and modulation in a buck converter is a dc modulation. In the ac signal modulation, if the input u of the PWM block is V_u , the output up and un in average mode will be: $V_{up} = V_u / (V_{pp} + V_{offset})$ $V_{un} = -V_{up}$ In this case, V_u is between $-(V_{pp} + V_{offset})$ and $V_{pp} + V_{offset}$, and V_{up} is between -1 to +1. In the dc signal modulation, the output up and un in average mode will be: $V_{up} = (V_u - V_{offset}) / V_{pp}$ $V_{un} = 1 - V_{up}$ In this case, V_u is between V_{offset} and $V_{pp} + V_{offset}$, and V_{up} is between 0 to +1. When it is set to the average mode, the PWM block outputs can be connected to a converter/inverter in the average mode model.

7.4.2 1-Phase PWM and 1-Phase PWM (phase shift)

The attributes for the **1-Phase PWM** block and **1-phase PWM (phase shift)** block are mostly the same. The difference is that the 1-Phase PWM block defines the phase shift through a parameter, while the 1-Phase PWM (phase shift) block reads the phase shift from an external input (labeled as "phase" in the image). The phase shift is in degree.

Images:



Attributes:

Parameters	Description
PWM Source	Source of the PWM generator. Without phase shift, it can be PWM 1 to PWM 7. With phase shift, it can be PWM 2 to PWM 7.
Output Mode	The output mode of the PWM generator. It can be one of the following: <ul style="list-style-type: none"> - <i>Use PWM A&B</i>: Both PWM outputs A and B are used, and they are complementary. - <i>Use PWM A</i>: Only PWM output A is used. - <i>Use PWM B</i>: Only PWM output B is used.
Dead Time	The dead time T_d for the PWM generator, in sec.
Sampling Frequency	Sampling frequency of the PWM generator, in Hz. The calculation is done and the PWM signal duty cycle is updated based on this frequency.
PWM Freq. Scaling Factor	The ratio between the PWM switching frequency and the sampling frequency. It can be from 1 to 100. For example, if the sampling frequency is 50 kHz and the scaling factor is 3, the PWM switching frequency will be 150 kHz. That is switches will operate at 150 kHz. But gating signals will be updated at 50 kHz, or once per 3 switching cycles.
Carrier Wave Type	The carrier wave type and the initial PWM output state. It can be one of the following: <ul style="list-style-type: none"> - <i>Triangular (start low)</i>: Triangular wave, and the initial PWM output state is low. - <i>Triangular (start high)</i>: Triangular wave, and the initial output state is high. - <i>Sawtooth (start low)</i>: Sawtooth wave, and the initial output state is low. - <i>Sawtooth (start high)</i>: Sawtooth wave, and the initial output state is high.
Trigger ADC	Setting whether for the PWM generator to trigger the A/D converter. It can be one of the following: <ul style="list-style-type: none"> - <i>Do not trigger ADC</i>: PWM does not trigger the A/D converter. - <i>Trigger ADC Group A</i>: PWM will trigger Group A of the A/D converter. - <i>Trigger ADC Group B</i>: PWM will trigger Group B of the A/D converter. - <i>Trigger ADC Group A&B</i>: PWM will trigger both Group A and B of the A/D converter.
ADC Trigger Position	The A/D trigger position ranges from 0 to a value less than 1. When it is 0, the A/D converter is triggered at the beginning of the PWM cycle, and when it is 0.5, the A/D converter is triggered at the 180° position of the PWM cycle.
Use Trip-Zone i	Define whether the PWM generator uses the i_{th} trip-zone signal or not, where i ranges from 1 to 6. It can be one of the following: <ul style="list-style-type: none"> - <i>Disable Trip-Zone i</i>: Disable the i_{th} trip-zone signal. - <i>One shot</i>: The PWM generator uses the trip-zone signal in the one-shot mode. Once triggered, the PWM must be started manually. - <i>Cycle by cycle</i>: The PWM generator uses the trip-zone signal in the cycle-by-cycle basis. The trip-zone signal is effective within the current cycle, and PWM will automatically re-start in the next cycle.

PWMA DC Trip Src1(DCAH)	Digital compare (DC) trip source DCAH for PWMA. The PWM channel may have up to two DC trip sources: DCAH and DCAL. The trip source can be one of the following: <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>Trip-zone 1</i>: PWM uses trip-zone 1 as digital compare input signal. - <i>Trip-zone 2</i>: PWM uses trip-zone 2 as digital compare input signal. - <i>Trip-zone 3</i>: PWM uses trip-zone 3 as digital compare input signal. - <i>Comparator 1</i>: PWM uses Comparator 1 as digital compare input signal. - <i>Comparator 2</i>: PWM uses Comparator 2 as digital compare input signal. - <i>Comparator 3</i>: PWM uses Comparator 3 as digital compare input signal.
PWMA DC Trip Src1(DCAL)	Digital compare (DC) trip source DCAL for PWMA. The trip source can be one of the following: <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>Trip-zone 1</i>: PWM uses trip-zone 1 as digital compare input signal. - <i>Trip-zone 2</i>: PWM uses trip-zone 2 as digital compare input signal. - <i>Trip-zone 3</i>: PWM uses trip-zone 3 as digital compare input signal. - <i>Comparator 1</i>: PWM uses Comparator 1 as digital compare input signal. - <i>Comparator 2</i>: PWM uses Comparator 2 as digital compare input signal. - <i>Comparator 3</i>: PWM uses Comparator 3 as digital compare input signal.
PWMA 1-shot Evt (DCAEVT1)	Define how the one-shot signal is used for the DC trip signal of PWMA. The active level of the DC trip signal can be selected from the following: <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>Trip source 1 is low</i>: PWM is tripped if the source signal 1 is low. - <i>Trip source 1 is high</i>: PWM is tripped if the source signal 1 is high. - <i>Trip source 2 is low</i>: PWM is tripped if the source signal 2 is low. - <i>Trip source 3 is high</i>: PWM is tripped if the source signal 2 is high. - <i>Trip source 1 low & source 2 high</i>: PWM is tripped if the source 1 signal is low and at the same time source 2 signal is high.
PWMA CBC Evt (DCAEVT1)	Define how the cycle-by-cycle signal is used for the DC trip signal of PWMA. The active level of the DC trip signal can be selected from the following: <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>Trip source 1 is low</i>: PWM is tripped if the source signal 1 is low. - <i>Trip source 1 is high</i>: PWM is tripped if the source signal 1 is high. - <i>Trip source 2 is low</i>: PWM is tripped if the source signal 2 is low. - <i>Trip source 3 is high</i>: PWM is tripped if the source signal 2 is high. - <i>Trip source 1 low & source 2 high</i>: PWM is tripped if the source 1 signal is low and at the same time source 2 signal is high.
PWMB DC Trip Src1(DCBH)	Digital compare (DC) trip source DCBH for PWMB. The PWM channel may have up to two DC trip sources: DCBH and DCBL. The trip source can be one of the following: <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>Trip-zone 1</i>: PWM uses trip-zone 1 as digital compare input signal. - <i>Trip-zone 2</i>: PWM uses trip-zone 2 as digital compare input signal. - <i>Trip-zone 3</i>: PWM uses trip-zone 3 as digital compare input signal. - <i>Comparator 1</i>: PWM uses Comparator 1 as digital compare input signal. - <i>Comparator 2</i>: PWM uses Comparator 2 as digital compare input signal. - <i>Comparator 3</i>: PWM uses Comparator 3 as digital compare input signal.

PWMB DC Trip Src1(DCBL)	Digital compare (DC) trip source DCBL for PWMB. The trip source can be one of the following: <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>Trip-zone 1</i>: PWM uses trip-zone 1 as digital compare input signal. - <i>Trip-zone 2</i>: PWM uses trip-zone 2 as digital compare input signal. - <i>Trip-zone 3</i>: PWM uses trip-zone 3 as digital compare input signal. - <i>Comparator 1</i>: PWM uses Comparator 1 as digital compare input signal. - <i>Comparator 2</i>: PWM uses Comparator 2 as digital compare input signal. - <i>Comparator 3</i>: PWM uses Comparator 3 as digital compare input signal.
PWMB 1-shot Evt (DCBEVT1)	Define how the one-shot signal is used for the DC trip signal of PWMB. The active level of the DC trip signal can be selected from the following: <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>Trip source 1 is low</i>: PWM is tripped if the source signal 1 is low. - <i>Trip source 1 is high</i>: PWM is tripped if the source signal 1 is high. - <i>Trip source 2 is low</i>: PWM is tripped if the source signal 2 is low. - <i>Trip source 3 is high</i>: PWM is tripped if the source signal 2 is high. - <i>Trip source 1 low & source 2 high</i>: PWM is tripped if the source 1 signal is low and at the same time source 2 signal is high.
PWMB CBC Evt (DCBEVT1)	Define how the cycle-by-cycle signal is used for the DC trip signal of PWMB. The active level of the DC trip signal can be selected from the following: <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>Trip source 1 is low</i>: PWM is tripped if the source signal 1 is low. - <i>Trip source 1 is high</i>: PWM is tripped if the source signal 1 is high. - <i>Trip source 2 is low</i>: PWM is tripped if the source signal 2 is low. - <i>Trip source 3 is high</i>: PWM is tripped if the source signal 2 is high. - <i>Trip source 1 low & source 2 high</i>: PWM is tripped if the source 1 signal is low and at the same time source 2 signal is high.
DC Event Filter Source	Source of the digital compare event filter. It can be one of the following: <ul style="list-style-type: none"> - <i>Do not use</i>: Do not use DC event filter. - <i>DCAEVT1</i>: Use DCAEVT1 as the filter source. - <i>DCAEVT2</i>: Use DCAEVT2 as the filter source. - <i>DCBEVT1</i>: Use DCBEVT1 as the filter source. - <i>DCBEVT2</i>: Use DCBEVT2 as the filter source. - <i>DCAEVT2</i>: PWM is tripped if the source signal 1 is high.
Blanking Window Pos (us)	Blanking window start position in a PWM period, in us.
Blanking Window Width (us)	Width of the blanking window, in us. The width is limited by the hardware, and can be calculated as below: $255 / \text{CPU Frequency}$ <p>For example, when the CPU speed is 90MHz, the width range is 0 to 2.83us.</p>
Blanking Window Range	Specify how the blanking window is applied. It can be one of the following: <ul style="list-style-type: none"> - <i>In the window</i>: The blanking action is applied with the window defined (from the start position for a window width defined) - <i>Out of window</i>: The blanking action is applied outside the window defined.
Applying Event Filtering	Specify how event filtering is applied to digital compare events. The event filtering can be applied to any combinations of the following digital compare events: DCAEVT1, DCAEVT2, DCBEVT1, and DCBEVT2

Trip Action	Define how the PWM generator responds to the trip action. It can be one of the following: <ul style="list-style-type: none"> - <i>High impedance</i>: PWM outputs in high impedance - <i>PWM A high & B low</i>: Set PWM A high and B low. - <i>PWM A low & B high</i>: Set PWM A low and B high. - <i>No action</i>: No action taken.
Peak-to-Peak Value	Peak-to-peak value V_{pp} of the carrier wave
Offset Value	DC offset value V_{offset} of the carrier wave
Phase Shift	Phase shift of the output with respect to the reference PWM generator output, in deg. (for 1-phase PWM generator without external phase shift input)
Initial Input Value	Initial value of the input
Use HRPWM	Define the high-resolution PWM. It can be one of the following: <ul style="list-style-type: none"> - <i>Do not use HRPWM</i>: Do not use high-resolution PWM - <i>Use HRPWM without calibration</i>: Use high-resolution PWM without calibration - <i>Use HRPWM with calibration</i>: Use high-resolution PWM with calibration
Start PWM at Beginning	When it is set to <i>Start</i> , PWM will start right from the beginning. If it is set to <i>Do not start</i> , one needs to start PWM using the Start PWM block.
Simulation Output Mode	The simulation output mode can be set to <i>Switching mode</i> or <i>Average mode</i> . When it is set to "Switching mode", the outputs of the PWM block are PWM signals. When it is set to "Average mode", the outputs of the PWM block are average mode signals. In the average mode, if the carrier wave is from negative to positive, and the absolute values of the negative peak and the positive peak are equal (for example, the carrier wave is from -1 to +1, or from -5 to +5), the modulation is considered as an ac signal modulation. Otherwise, the modulation is considered as a dc signal modulation. For example, modulation in a 3-phase or single-phase inverter is an ac modulation, and modulation in a buck converter is a dc modulation. In the ac signal modulation, if the input of the PWM block is V_{in} , the output A and B in average mode will be: $V_A = V_{in} / (V_{pp} + V_{offset})$ $V_B = -V_A$ In this case, V_{in} is between $-(V_{pp} + V_{offset})$ and $V_{pp} + V_{offset}$, and V_A and V_B are between -1 to +1. In the dc signal modulation, the output A and B in average mode will be: $V_A = (V_{in} - V_{offset}) / V_{pp}$ $V_B = 1 - V_A$ In this case, V_{in} is between V_{offset} and $V_{pp} + V_{offset}$, and V_A and V_B are between 0 to +1. When it is set to the average mode, the PWM block outputs can be connected to a converter/inverter in the average mode model.

Phase Shift

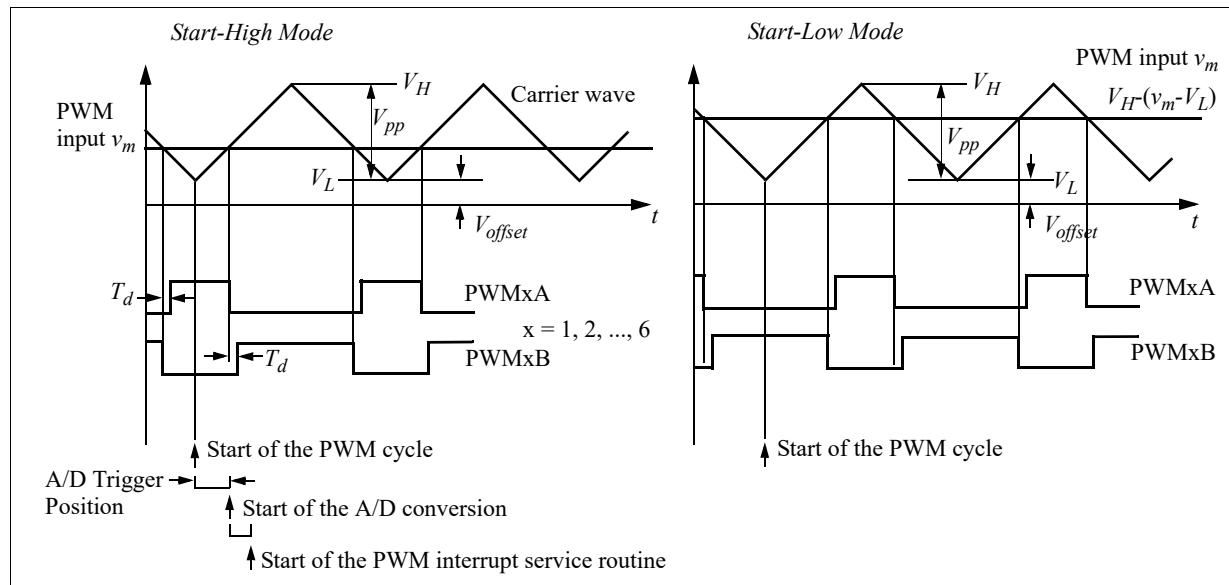
A 1-phase PWM generator can generate PWM signal that is phase shifted with respect to another PWM signal. The way how PWM blocks are defined for phase shift is explained in Section 7.4.5.

The phase shift value is in degrees. When the value is -30° , the output is shifted to the right (lagging) by 30° of the switching cycle with respect to the reference PWM generator output. This is equivalent to shifting the PWM carrier wave to the right by 30° . When the phase value is 30° , the output is shifted to the left (leading) by 30° .

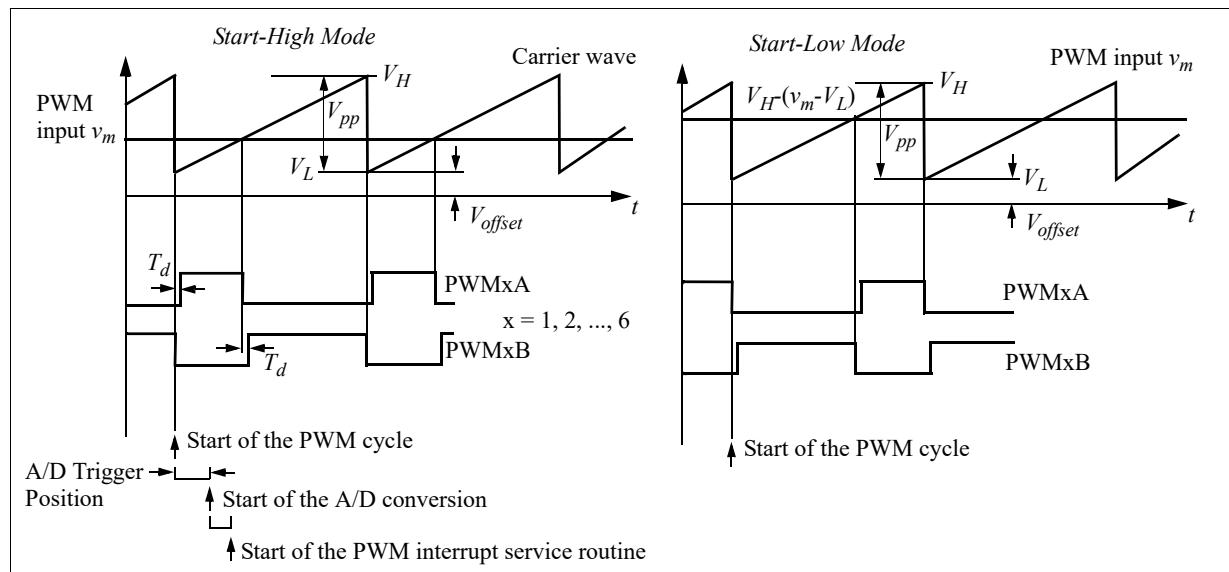
Carrier Wave

There are two types of carrier waveforms: triangular wave (with equal rising and falling slope intervals) and sawtooth wave. In addition, there are two operation modes: start-low and start-high modes, as explained below.

The input and output waveforms of a PWM generator with the triangular carrier wave are shown below:



The input and output waveforms of a PWM generator with the sawtooth carrier wave are shown below:



The figures above show how the dead time is defined, and the time sequence when the PWM generator triggers the A/D converter. If triggering the A/D converter is selected, from the start of the PWM cycle, after a certain delay defined by the A/D trigger position, the A/D conversion will start. After the A/D conversion is completed, the PWM interrupt service routine will start.

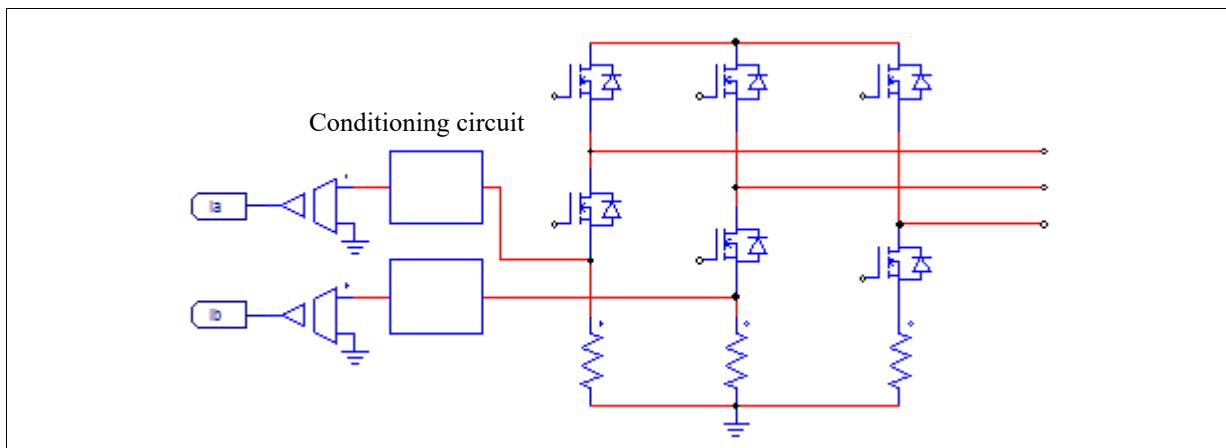
If the PWM generator does not trigger the A/D converter, the PWM interrupt service routine will start at the beginning of the PWM cycle.

The figures above also show how the start-high and start-low modes work. Assume that the PWM input is v_m , and the lowest value of the carrier wave is V_L and the highest value is V_H . In the start-high mode, the PWM positive output PWMA is high at the beginning of the switching cycle, and it remains high as long as the input v_m is greater than the carrier wave. For example, for a carrier wave from 0 to 1, $V_L=0$, and $V_H=1$. If $v_m=0.2$, the PWM output PWMA will remain high as long as the carrier is less than 0.2.

On the other hand, in the start-low mode, the PWM positive output PWMA is low at the beginning of the switching cycle, and it is high when the carrier wave is greater than the value $V_H(v_m-V_L)$. For example, for a carrier wave from 0 to 1, $V_L=0$, and $V_H=1$. If $v_m=0.2$, the PWM output PWMA will be high as long as the carrier is greater than 0.8.

The carrier start mode depends on how switch currents are measured. In a 3-phase inverter, for example, if top switch currents are measured, the start-high mode should be selected. This ensures that at the beginning of the cycle, the top switch gating signal is high and the current is conducting. On the other hand, if bottom switch currents are measured, the start-low mode should be selected. This ensures that at the beginning of the cycle, the top switch gating signal is low, and the bottom switch gating signal is high and the current is conducting.

For example, in the circuit below, the bottom switch currents of Phase A and B are measured. In this case, the carrier start-low mode should be selected.

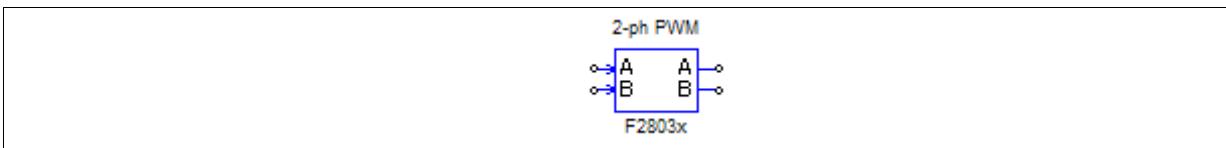


Note: In the start-low mode, the PWM input v_m is converted to $V_H(v_m-V_L)$ internally before it is compared with the carrier wave to generate the PWM signal. With the conversion, both the start-low and start-high modes will have the same duty cycle expression. For example, for a sawtooth wave with $V_L=0$ and $V_H=1$, or for a triangular wave with $V_L= -V_H$, the duty cycle D of the PWMA output in both the start-low and start-high modes is: $D = v_m/V_H$.

7.4.3 2-Phase PWM

A 2-phase PWM block operates in one of 6 operation modes.

Image:



Attributes:

Parameters	Description
PWM Source	Source of the PWM generator. It can be PWM 1 to PWM 6.
Mode Type	The operation mode of the PWM generation. It can be one of the 6 modes. The waveforms of the 6 operation modes are described below.
Sampling Frequency	Sampling frequency of the PWM generator, in Hz. The calculation is done and the PWM signal duty cycle is updated based on this frequency.
PWM Freq. Scaling Factor	The ratio between the PWM switching frequency and the sampling frequency. It can be from 1 to 100. For example, if the sampling frequency is 50 kHz and the scaling factor is 3, the PWM switching frequency will be 150 kHz. That is switches will operate at 150 kHz. But gating signals will be updated at 50 kHz, or once per 3 switching cycles.
Trigger ADC	Setting whether for the PWM generator to trigger the A/D converter. It can be one of the following: <ul style="list-style-type: none"> - <i>Do not trigger ADC</i>: PWM does not trigger the A/D converter. - <i>Trigger ADC Group A</i>: PWM will trigger Group A of the A/D converter. - <i>Trigger ADC Group B</i>: PWM will trigger Group B of the A/D converter. - <i>Trigger ADC Group A&B</i>: PWM will trigger both Group A and B of the A/D converter.
ADC Trigger Position	The A/D trigger position ranges from 0 to a value less than 1. When it is 0, the A/D converter is triggered at the beginning of the PWM cycle, and when it is 0.5, the A/D converter is triggered at the 180° position of the PWM cycle.
Use Trip-Zone <i>i</i>	Define whether the PWM generator uses the <i>i</i> th trip-zone signal or not, where <i>i</i> ranges from 1 to 6. It can be one of the following: <ul style="list-style-type: none"> - <i>Disable Trip-Zone <i>i</i></i>: Disable the <i>i</i>th trip-zone signal. - <i>One shot</i>: The PWM generator uses the trip-zone signal in the one-shot mode. Once triggered, the PWM must be started manually. - <i>Cycle by cycle</i>: The PWM generator uses the trip-zone signal in the cycle-by-cycle basis. The trip-zone signal is effective within the current cycle, and PWM will automatically re-start in the next cycle.
PWMA DC Trip Src1(DCAH)	Digital compare (DC) trip source DCAH for PWMA. The PWM channel may have up to two DC trip sources: DCAH and DCAL. The trip source can be one of the following: <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>Trip-zone 1</i>: PWM uses trip-zone 1 as digital compare input signal. - <i>Trip-zone 2</i>: PWM uses trip-zone 2 as digital compare input signal. - <i>Trip-zone 3</i>: PWM uses trip-zone 3 as digital compare input signal. - <i>Comparator 1</i>: PWM uses Comparator 1 as digital compare input signal. - <i>Comparator 2</i>: PWM uses Comparator 2 as digital compare input signal. - <i>Comparator 3</i>: PWM uses Comparator 3 as digital compare input signal.
PWMA DC Trip Src1(DCAL)	Digital compare (DC) trip source DCAL for PWMA. The trip source can be one of the following: <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>Trip-zone 1</i>: PWM uses trip-zone 1 as digital compare input signal. - <i>Trip-zone 2</i>: PWM uses trip-zone 2 as digital compare input signal. - <i>Trip-zone 3</i>: PWM uses trip-zone 3 as digital compare input signal. - <i>Comparator 1</i>: PWM uses Comparator 1 as digital compare input signal. - <i>Comparator 2</i>: PWM uses Comparator 2 as digital compare input signal. - <i>Comparator 3</i>: PWM uses Comparator 3 as digital compare input signal.

PWMA 1-shot Evt (DCAEVT1)	Define how the one-shot signal is used for the DC trip signal of PWMA. The active level of the DC trip signal can be selected from the following: <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>Trip source 1 is low</i>: PWM is tripped if the source signal 1 is low. - <i>Trip source 1 is high</i>: PWM is tripped if the source signal 1 is high. - <i>Trip source 2 is low</i>: PWM is tripped if the source signal 2 is low. - <i>Trip source 3 is high</i>: PWM is tripped if the source signal 2 is high. - <i>Trip source 1 low & source 2 high</i>: PWM is tripped if the source 1 signal is low and at the same time source 2 signal is high.
PWMA CBC Evt (DCAEVT1)	Define how the cycle-by-cycle signal is used for the DC trip signal of PWMA. The active level of the DC trip signal can be selected from the following: <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>Trip source 1 is low</i>: PWM is tripped if the source signal 1 is low. - <i>Trip source 1 is high</i>: PWM is tripped if the source signal 1 is high. - <i>Trip source 2 is low</i>: PWM is tripped if the source signal 2 is low. - <i>Trip source 3 is high</i>: PWM is tripped if the source signal 2 is high. - <i>Trip source 1 low & source 2 high</i>: PWM is tripped if the source 1 signal is low and at the same time source 2 signal is high.
PWMB DC Trip Src1(DCBH)	Digital compare (DC) trip source DCBH for PWMB. The PWM channel may have up to two DC trip sources: DCBH and DCBL. The trip source can be one of the following: <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>Trip-zone 1</i>: PWM uses trip-zone 1 as digital compare input signal. - <i>Trip-zone 2</i>: PWM uses trip-zone 2 as digital compare input signal. - <i>Trip-zone 3</i>: PWM uses trip-zone 3 as digital compare input signal. - <i>Comparator 1</i>: PWM uses Comparator 1 as digital compare input signal. - <i>Comparator 2</i>: PWM uses Comparator 2 as digital compare input signal. - <i>Comparator 3</i>: PWM uses Comparator 3 as digital compare input signal.
PWMB DC Trip Src1(DCBL)	Digital compare (DC) trip source DCBL for PWMB. The trip source can be one of the following: <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>Trip-zone 1</i>: PWM uses trip-zone 1 as digital compare input signal. - <i>Trip-zone 2</i>: PWM uses trip-zone 2 as digital compare input signal. - <i>Trip-zone 3</i>: PWM uses trip-zone 3 as digital compare input signal. - <i>Comparator 1</i>: PWM uses Comparator 1 as digital compare input signal. - <i>Comparator 2</i>: PWM uses Comparator 2 as digital compare input signal. - <i>Comparator 3</i>: PWM uses Comparator 3 as digital compare input signal.
PWMB 1-shot Evt (DCBEVT1)	Define how the one-shot signal is used for the DC trip signal of PWMB. The active level of the DC trip signal can be selected from the following: <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>Trip source 1 is low</i>: PWM is tripped if the source signal 1 is low. - <i>Trip source 1 is high</i>: PWM is tripped if the source signal 1 is high. - <i>Trip source 2 is low</i>: PWM is tripped if the source signal 2 is low. - <i>Trip source 3 is high</i>: PWM is tripped if the source signal 2 is high. - <i>Trip source 1 low & source 2 high</i>: PWM is tripped if the source 1 signal is low and at the same time source 2 signal is high.

PWMB CBC Evt (DCBEVT1)	Define how the cycle-by-cycle signal is used for the DC trip signal of PWMB. The active level of the DC trip signal can be selected from the following: <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>Trip source 1 is low</i>: PWM is tripped if the source signal 1 is low. - <i>Trip source 1 is high</i>: PWM is tripped if the source signal 1 is high. - <i>Trip source 2 is low</i>: PWM is tripped if the source signal 2 is low. - <i>Trip source 3 is high</i>: PWM is tripped if the source signal 2 is high. - <i>Trip source 1 low & source 2 high</i>: PWM is tripped if the source 1 signal is low and at the same time source 2 signal is high.
DC Event Filter Source	Source of the digital compare event filter. It can be one of the following: <ul style="list-style-type: none"> - <i>Do not use</i>: Do not use DC event filter. - <i>DCAEVT1</i>: Use DCAEVT1 as the filter source. - <i>DCAEVT2</i>: Use DCAEVT2 as the filter source. - <i>DCBEVT1</i>: Use DCBEVT1 as the filter source. - <i>DCBEVT2</i>: Use DCBEVT2 as the filter source. - <i>DCAEVT2</i>: PWM is tripped if the source signal 1 is high.
Blanking Window Pos (us)	Blanking window start position in a PWM period, in us.
Blanking Window Width (us)	Width of the blanking window, in us. The width is limited by the hardware, and can be calculated as below: $255 / \text{CPU Frequency}$ <p>For example, when the CPU speed is 90MHz, the width range is 0 to 2.83us.</p>
Blanking Window Range	Specify how the blanking window is applied. It can be one of the following: <ul style="list-style-type: none"> - <i>In the window</i>: The blanking action is applied with the window defined (from the start position for a window width defined) - <i>Out of window</i>: The blanking action is applied outside the window defined.
Applying Event Filtering	Specify how event filtering is applied to digital compare events. The event filtering can be applied to any combinations of the following digital compare events: DCAEVT1, DCAEVT2, DCBEVT1, and DCBEVT2
Trip Action	Define how the PWM generator responds to the trip action. It can be one of the following: <ul style="list-style-type: none"> - <i>High impedance</i>: PWM outputs in high impedance - <i>PWM A high & B low</i>: Set PWM A high and B low. - <i>PWM A low & B high</i>: Set PWM A low and B high. - <i>No action</i>: No action taken.
Peak Value	Peak value V_{pk} of the carrier wave
Initial Input Value A, B	Initial value of the inputs A and B.
Start PWM at Beginning	When it is set to "Start", PWM will start right from the beginning. If it is set to "Do not start", one needs to start PWM using the "Start PWM" function.

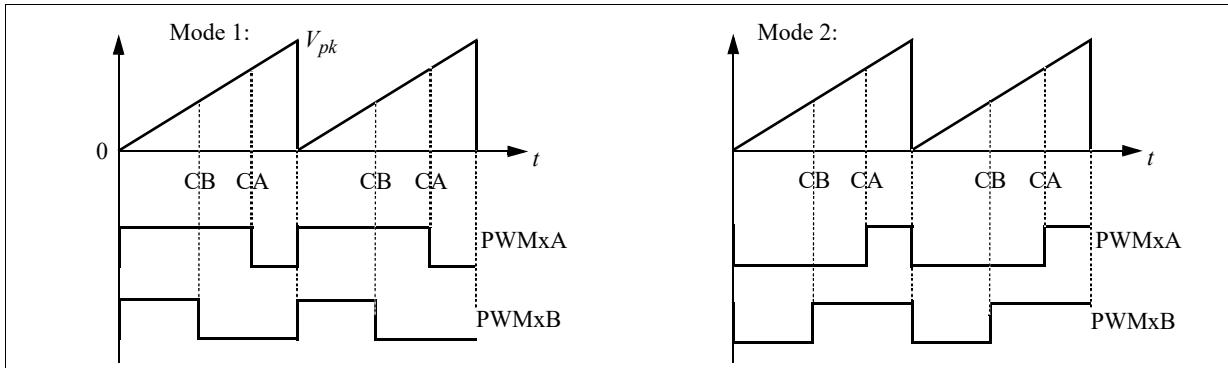
For 2-phase PWM generators, the outputs are determined based on the mode of operation, as described below. The carrier wave is either sawtooth or triangular, depending on the mode of operation. It increases from 0 to the peak value V_{pk} , and there is no dc offset.

Operation Mode 1:

The figure below on the left shows the waveforms of Mode 1. In the figure, "CA" and "CB" refer to two inputs A and B of the 2-phase PWM generator. Each input controls the turn-off time of each output.

Operation Mode 2:

The figure below on the right shows the waveforms of Mode 2. Unlike in Mode 1, each input controls the turn-on time of each output.

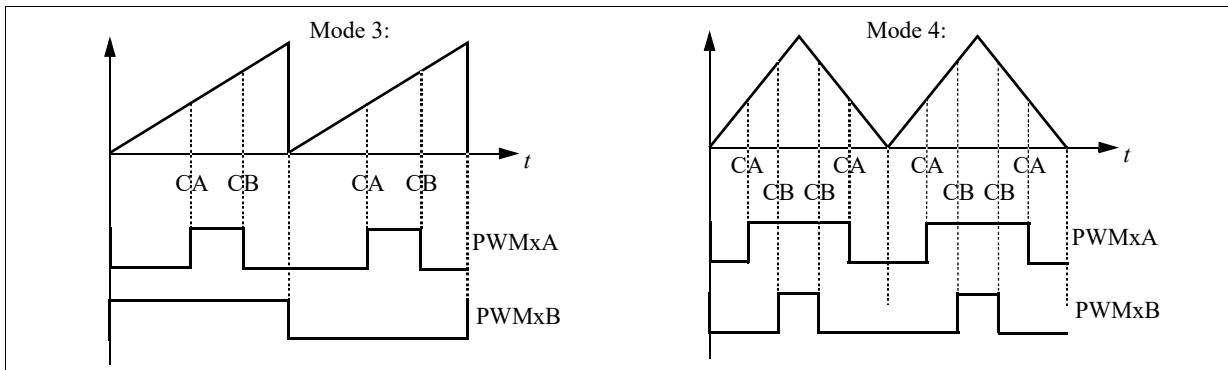


Operation Mode 3:

The figure below on the left shows the waveforms of Mode 3. Input A controls the turn-on and Input B controls the turn-off of the PWM output A. The PWM output B is on for one complete PWM cycle, and is off for the next cycle.

Operation Mode 4:

The figure below on the right shows the waveforms of Mode 4. The carrier wave is triangular. Each input controls both the turn-on and turn-off of its output.

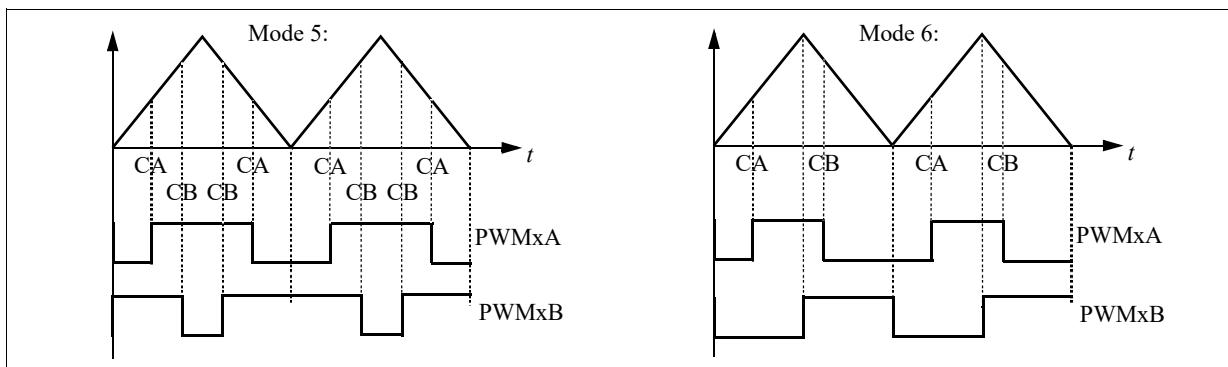


Operation Mode 5:

The figure below on the left shows the waveforms of Mode 5. The carrier wave is triangular. Similar to Mode 4, each input controls both the turn-on and turn-off of its output. Note that PWM output B is inverted in this case.

Operation Mode 6:

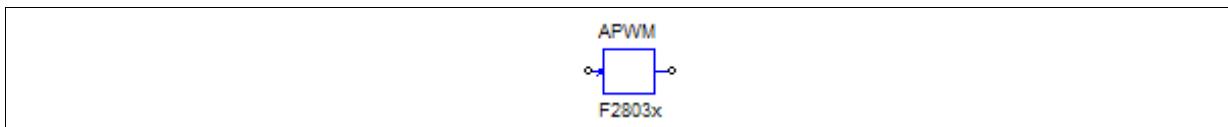
The figure below on the right shows the waveforms of Mode 6. In this mode, Input A controls the turn-on and Input B controls the turn-off of PWM output A. The PWM output B is on for the first half PWM cycle, and is off for the second half cycle.



7.4.4 Single PWM (shared with capture)

A single PWM generator, also called APWM, shares the same resource as captures.

Image:



Attributes:

Parameters	Description
PWM Source	APWM generators share the same resource as captures. The PWM source can be from one of the three designated GPIO ports: <i>APWM 1 (GPIO5, 19, 24)</i>
PWM Frequency	Frequency of the PWM generator, in Hz
Carrier Wave Type	The carrier wave type and the initial PWM output state. It can be one of the following: - <i>Sawtooth (start low)</i> : Sawtooth wave, with the PWM output in the low state initially. - <i>Sawtooth (start high)</i> : Sawtooth wave, with the PWM output in the high state initially.
Stop Action	The output status when the PWM generator is stopped. It can be one of the following: - <i>Output low</i> : The PWM output will be set to low. - <i>Output high</i> : The PWM output will be set to high.
Peak-to-Peak Value	Peak-to-peak value of the carrier wave
Offset Value	DC offset value of the carrier wave
Phase Shift	Phase shift of the output with respect to the reference PWM generator, in deg.
Initial Input Value	Initial value of the input
Start PWM at Beginning	When it is set to <i>Start</i> , PWM will start right from the beginning. If it is set to <i>Do not start</i> , one needs to start PWM using the Start PWM block.

Similar to 1-phase PWM generators, an APWM generator can generate a PWM signal that has a phase shift with respect to another PWM generator. The way how PWM blocks are defined for phase shift is explained in Section 7.4.5.

As noted before, the APWM generators has reduced number of functions than 1-phase PWM generators. It can not trigger the A/D converter and can not use the trip-zone signal.

7.4.5 Synchronization Between PWM Blocks

Three types of PWM blocks can be synchronized, and phase shifts can be defined between each other: **1-phase PWM**, **1-phase PWM (phase shift)**, and **APWM (or Single PWM (shared with capture))**.

A 1-phase PWM block can generate PWM signal that is phase shifted with respect to another PWM signal. There is one series in regular PWM blocks: PWM 1, 2, 3, 4, 5, 6, and 7.

The definitions of the PWM blocks for phase shift are described below.

- The reference PWM and the PWM being phase shifted must be from the same series. That is, PWM 1 can be the reference, and PWM 2, 3, 4, 5, 6, and 7 can be phase shifted with respect to PWM 1. Or PWM 2 can be the reference, and PWM 3 to 7 can be phase shifted with respect to PWM 2.
- APWM 1 can be phase shifted with respect to PWM 1.
- The reference PWM and the PWM being shifted must be consecutive in the series, unless the skipped PWM is not used. For example, if PWM 1, 2, and 3 are all used, but PWM 2 is not synchronized with

PWM 1 and 3, this is not allowed. However, if PWM 2 is not used in the circuit, it is ok to use PWM 1 as the reference and phase shift PWM 3.

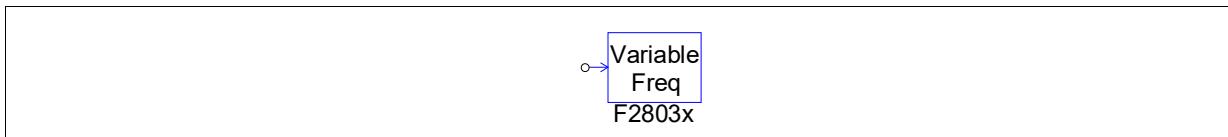
For example, the following definitions are correct, with the first PWM as the reference and the subsequent PWM blocks phase shifted:

PWM 1 (reference), PWM 2, PWM 3, PWM 4, PWM 5, PWM 6, PWM 7
 PWM 2 (reference), PWM 3
 PWM 4 (reference), PWM 5
 PWM 5 (reference), PWM 6
 PWM 6 (reference), PWM 7
 PWM 1 (reference), PWM 2, PWM 3, PWM 4, PWM 5, PWM 6, PWM 7, APWM 1
 PWM 1 (reference), APWM 1

7.5 Variable Frequency PWM

The Variable Frequency PWM block provides the function to change the sampling frequency of a PWM generator. The image and parameters are shown below.

Image:



Attributes:

Parameters	Description
PWM Source	Source of the PWM generator. It can be one of the following: PWM 1, PWM 2, PWM 3, PWM 4, PWM 5, PWM 6, 3-phase PWM 123, and 3-phase PWM 456.
Adjust Interrupt Pos.	Specify if the interrupt position is adjusted with the frequency. It can be one of the following: <ul style="list-style-type: none"> - <i>Do not adjust</i>: The interrupt position will remain unchanged as calculated with the base frequency. - <i>Adjust</i>: The interrupt position will be recalculated at the beginning of each cycle based on the new frequency.
Adjust Ramp Compensation	Specify if the ramp compensation of the comparator DAC block is adjusted with the frequency. It can be one of the following: <ul style="list-style-type: none"> - <i>Do not adjust</i>: The ramp compensation will remain unchanged as calculated with the base frequency. - <i>Adjust</i>: The ramp compensation will be recalculated at the beginning of each cycle based on the new frequency.

The sampling frequency of the corresponding PWM block will be changed at the beginning of the next PWM period as follows:

$$\text{PWM_Frequency} = \text{PWM_Base Frequency} / \text{Input_Value}$$

where *PWM_Base_Frequency* is the sampling frequency of the corresponding PWM block, and *Input_Value* is the input value of this block.

If the interrupt position is to be adjusted, the interrupt position will be recalculated in each cycle. Since adjusting the interrupt position takes time, if the frequency change is small, it is recommended not to adjust the interrupt position.

Similarly, if the ramp compensation is to be adjusted, the ramp compensation will be recalculated in each cycle. Since adjusting the ramp compensation takes time, if the frequency change is small, it is recommended not to

adjust the ramp compensation.

7.6 Start PWM and Stop PWM

The Start PWM and Stop PWM blocks provide the function to start/stop a PWM generator. The images and parameters are shown below.

Images:



Attributes:

Parameters	Description
PWM Source	The source of the PWM generator. It can be: PWM 1-7, 3-phase PWM 123 and PWM 456, and Capture 1.

7.7 Trip-Zone and Trip-Zone State

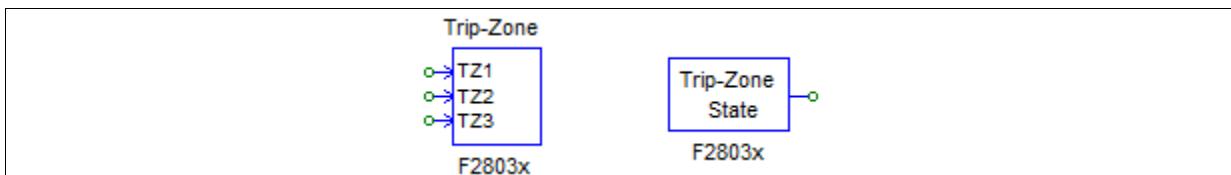
F2803x contains 6 trip-zone input signals, 3 from GPIO ports and 3 from Comparators. Comparators can only be used in Digital Compare to trip PWM.

Trip-zone is used to handle external fault or trip conditions. The corresponding PWM outputs can be programmed to respond accordingly.

One trip-zone signal can be used by multiple PWM generators, and a PWM generator can use any or all of the 6 trip-zone signals. The interrupt generated by trip-zone signals are handled by the interrupt block.

The trip-zone signal triggers a trip action when the input signal is low (0). The trip-zone signals through Digital Compare trigger a trip action in specified level (either high or low)

Image:



Attributes for Trip-Zone:

Parameters	Description
Use Trip-Zone i	Specify if this Trip-Zone i is used.
GPIO Port for Trip-Zone i	Specify a designated GPIO port as Trip-Zone input signal. - GPIO port for trip-zone 1: select either GPIO12 or 13 - GPIO port for trip-zone 2: select either GPIO13, 16, or 18 - GPIO port for trip-zone 3: select either GPIO14, 17, or 19
Use Comparator i	Specify if comparator 1, 2, or 3 is used as Trip-Zone input signal i

Attributes for Trip-Zone State:

Parameters	Description
PWM Source	The source of the PWM generator. It can be: PWM 1-7, and 3-phase PWM 123 and PWM 456.

The trip-zone interrupt can be generated in either one-shot mode or cycle-by-cycle mode, as defined in the PWM generator parameter input. In the cycle-by-cycle mode, the interrupt only affects the PWM output within the current PWM cycle. On the other hand, in the one-shot mode, interrupt triggers a trip action when the input signal is low (0). This will set the PWM output permanently, and the PWM generator must be restarted to resume the operation.

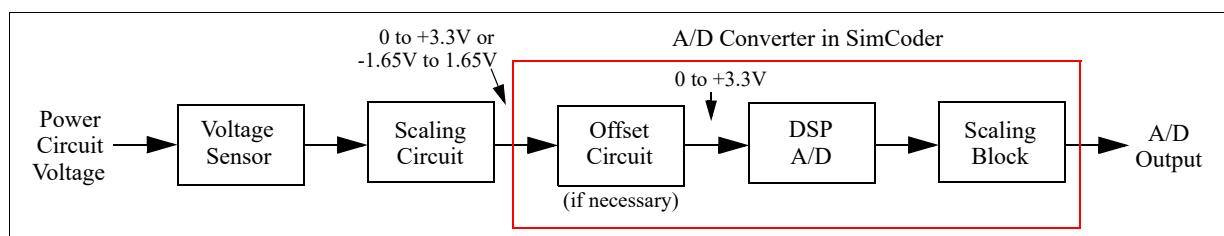
The Trip-Zone State element indicates whether the trip-zone signal is in one-shot mode or cycle-by-cycle mode when it triggers a PWM generator to generate an interrupt. When the output is 1, it means that the trip-zone signal is in one-shot mode. When the output is 0, the trip-zone signal is in cycle-by-cycle mode.

Note that when defining the interrupt block associate with trip-zone, the "Device Name" parameter of the interrupt block should be the name of the PWM generator, not the trip-zone block name. For example, if a PWM generator called "PWM_G1" uses trip-zone 1 in the trip-zone block "TZ1". The "Device Name" of the corresponding interrupt block should be "PWM_G1", not "TZ1". The "Channel Number" parameter in the interrupt block is not used in this case.

7.8 A/D Converter

F2803x provides a 16-channel 12-bit A/D converter.

Normally a power circuit quantity (voltage, current, speed, etc.) is brought to the DSP in several stages. For example, a power circuit voltage, which could be at a high level, is first converted to a control signal using a voltage sensor. A scaling circuit is then used to scale the signal, and an offset circuit is used to provide dc offset to the signal if necessary, so that the signal at the DSP A/D input is within the range of 0V and +3.3V. This signal is converted to a digital value in DSP, and a scaling block may be used to scale the value back to its original value. The complete process is shown in the diagram below.

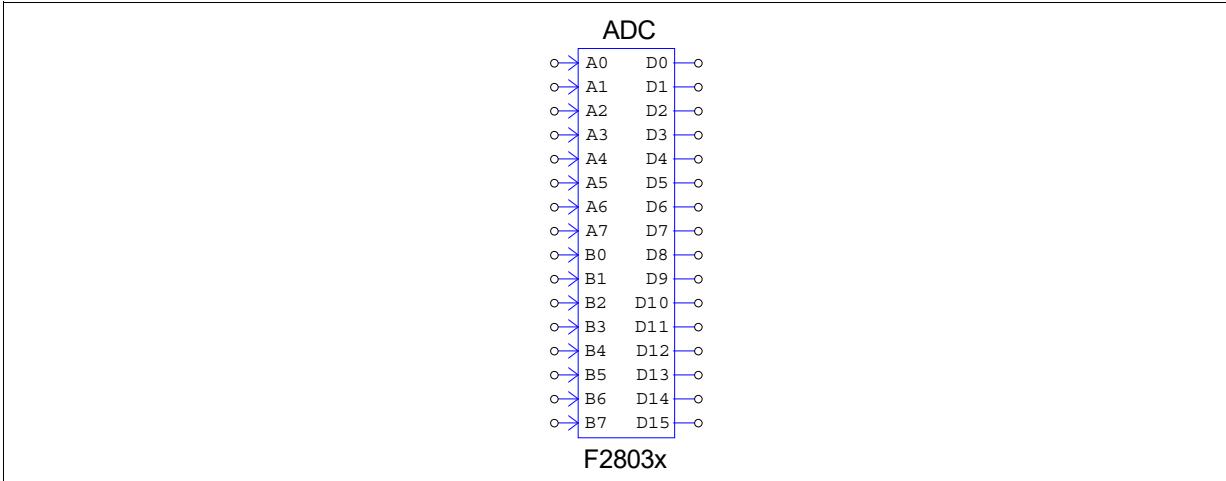


As shown above, the A/D converter element in SimCoder is not exactly the same as the physical A/D converter on the DSP. Rather, it combines the functions of an offset circuit, the DSP A/D converter, and a scaling block. This is designed for the convenience of AC system applications. It will be further explained in Section 6.5.3.

In many applications, the circuit variables to be monitored are AC signals, especially in AC motor drive systems. For each of this kind of AC signals, an offset circuit must be built in the hardware on circuit board at the input of the DSP analog input, in order to shift the signal level to the acceptable range of 0 to +3.3V. SimCoder's A/D converter provides the convenience for such cases. Instead of level-shifting and scaling the A/D output signals, user may chose to use the offset option and scaling factor in the SimCoder A/D converter, and the target code will be generated accordingly.

The image and the parameters of the A/D converter in the target library are described below. In the following description, "A/D converter" refers to the A/D converter in the target library, not the DSP A/D converter, unless otherwise stated.

Image:



Attributes:

Parameters	Description
Ch A_i or B_i Mode	Input mode of the i_{th} A/D converter channel A_i or B_i . The input mode can be one of the following: - <i>AC</i> : This option is for simulation only, not for code generation. The input range is considered from -1.65V to +1.65V. This option includes the offset circuit into the A/D converter. It provides the convenience in cases where an external level shifter is needed to shift the AC signal to the 0 to +3.3V range. - <i>DC</i> : The input is a dc value, and the range is from 0 to +3.3V.
Ch A_i or B_i Gain	Gain k of the i_{th} A/D converter channel A_i or B_i .
Conversion Order	Order of the A/D conversion. If the field is left blank (undefined), the conversion will be done based on the serial numbers of the A/D channel. For example, if A0, A2, A4, B1, and B3 are used, the conversion will be done in this order: A0, A2, A4, B1, and B3. If you wish certain channels to be performed first, you can define the order here. For example, if the conversion order is defined as: A4,A0,A2,B3,B2 The conversion will be done in the order defined, that is, A4 before A0, and A0 before A2, and so on.
ADCINT1 PIE Selection	Specify if interrupt ADCINT1 uses <i>PIE Group1</i> or <i>PIE Group10</i> .
ADCINT2 PIE Selection	Specify if interrupt ADCINT2 uses <i>PIE Group1</i> or <i>PIE Group10</i> .

An A/D converter has up to 16 channels. SimCoder divides them into groups according their sampling rates. The group with the highest sampling rate uses interrupt ADCINT1, and the group with the second highest sampling rate uses interrupt ADCINT2, etc. The two ADC groups with the highest sampling rates can choose interrupt from PIE (peripheral interrupt expansion) groups of the PIE vector table for different interrupt priority. PIE Group1 has a higher interrupt priority than PIE Group10. For example, PWM's interrupt is in PIE Group3, Its interrupt priority is lower than PIE Group1 but higher than Group10. If one wants PWM interrupt to have a higher priority than ADC interrupt, one needs to set the interrupt corresponding to the ADC channels to use PIE Group10.

Trigger Source:

The A/D converter can be triggered from multiple sources. Multiple A/D channels may share the same trigger source. Each A/D channel can be triggered by:

- One of the PWM generators,
- Timer1 or Timer2.
- More than one trigger source.

In a schematic, if a A/D channel is not associated with a PWM generator, one should insert a ZOH block at the output of the A/D channel so that SimCoder will select the timer as the trigger source.

It is not permitted to have a A/D converter channel triggered by one source, but its output signal is used in a circuit section that has a different sampling rate.

Output Scaling:

The output is scaled based on the following:

$$V_o = k * V_i$$

where V_i is the value at the input of the A/D converter.

Input Offset and Scaling:

The input of the A/D converter must stay within the input range. When the input is out of the range, it will be clamped to the limit, and a warning message will be given.

Also, the signal at the input port of the A/D converter must be scaled such that, when the input mode is DC, the maximum input voltage be scaled to +3.3V; and when the input mode is AC, the peak voltage be scaled to +/- 1.65V.

To illustrate how to use the A/D converter, two examples are given below: One with a dc input and the other with an ac input.

Assume that a power circuit voltage is a dc quantity, and the range is as follows:

$$V_{i_min} = 0 \text{ V}$$

$$V_{i_max} = 150 \text{ V}$$

The input mode of the A/D converter will be set to dc, and the input range is from 0 to +3.3V. Assume that the actual value of the voltage at a certain point is:

$$V_i = 100 \text{ V}$$

Let the voltage sensor gain be 0.01. After the voltage sensor, the maximum value and the actual value of the input become:

$$V_{i_max_s} = 150 * 0.01 = 1.5 \text{ V}$$

$$V_{i_s} = 100 * 0.01 = 1 \text{ V}$$

To utilize the full range of the DSP, a conditioning circuit with a gain of 2.2 will be used. The combined gain of the voltage sensor and the conditioning circuit becomes: $0.01 * 2.2 = 0.022$. After the conditioning circuit and at the input of the DSP A/D converter, the maximum value and the actual value of the input become:

$$V_{i_max_s_c} = 1.5 * 2.2 = 3.3 \text{ V}$$

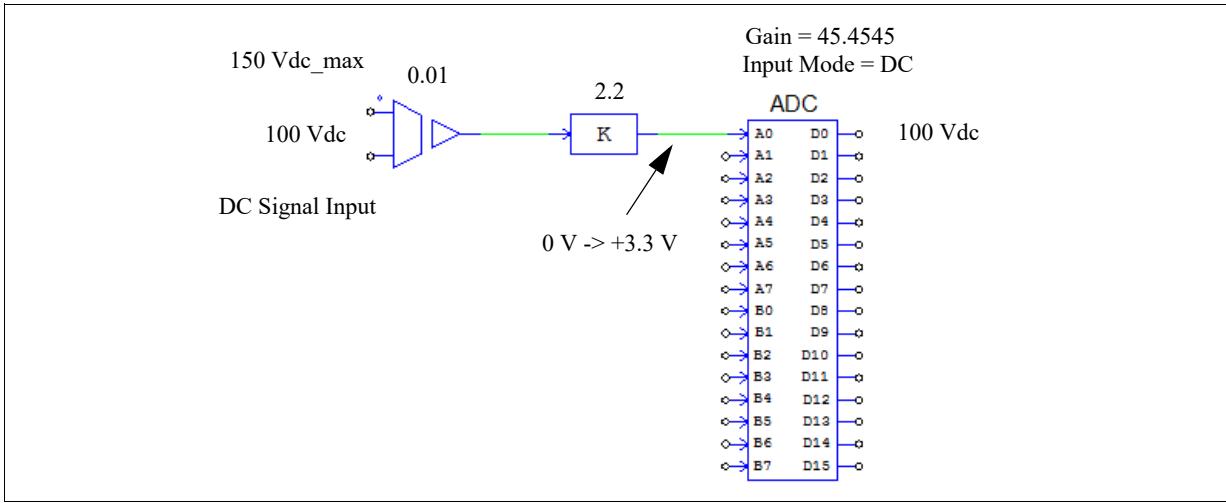
$$V_{i_s_c} = 1 * 2.2 = 2.2 \text{ V}$$

The scaling block after the DSP A/D can be selected such that the original power circuit quantity is restored. In this example, a gain of 45.4545 will be used. Note that this is the reciprocal of the combined gain of the voltage sensor and the conditioning circuit. At the A/D output, the maximum value and the actual value are:

$$V_{o_max} = 45.4545 * 3.3 = 150 \text{ V}$$

$$V_o = 45.4545 * 2.2 = 100 \text{ V}$$

The gain of the A/D channel will be set to 45.4545. The circuit connection and the settings are shown in the figure below.



Please note that, in this example, if the gain of the proportional block is changed from 2.2 to 1.1, and the A/D gain is changed from 45.4545 to 90.909, the simulation results will be the same. But the generated hardware code will not be correct. This is because the hardware code assumes that the maximum input value is scaled to +3.3V, but in this case it is only +1.5V. Therefore, one must set up the circuit such that, in the dc mode, the maximum input value is scaled to be +3.3V.

In another example, assume that a power circuit voltage is an ac quantity, and the range is as follows:

$$V_{i_max} = +/- 75 \text{ V}$$

The input mode of the A/D converter will be set to ac, and the input range is from -1.65V to +1.65V. Assume that the actual value of the voltage has a peak value of:

$$V_i = +/- 50 \text{ V}$$

Let the voltage sensor gain be 0.01. After the voltage sensor, the maximum value and the actual value of the input become:

$$V_{i_max_s} = +/- 0.75 \text{ V}$$

$$V_{i_s} = +/- 0.5 \text{ V}$$

Since the A/D converter input range is from -1.65V to +1.65V, this signal must be scaled before it is sent to the DSP. A conditioning circuit with a gain of 2.2 is needed (i.e. $1.65/0.75 = 2.2$). After the conditioning circuit and at the input of the DSP A/D converter, the maximum value and the actual value of the input become:

$$V_{i_max_s_c} = +/- 1.65 \text{ V}$$

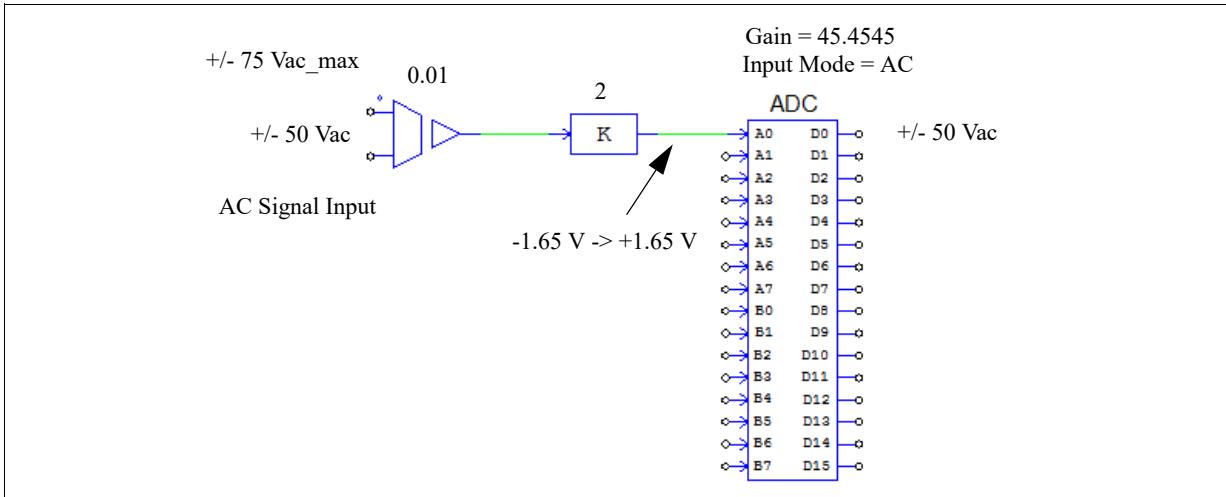
$$V_{i_s_c} = +/- 1.1 \text{ V}$$

The scaling block after the DSP A/D can be selected such that the original power circuit quantity is restored. In this example, a gain of 45.4545 will be used. Note that this is the reciprocal of the combined gain of the voltage sensor and the conditioning circuit. At the A/D output, the maximum value and the actual value are:

$$V_{o_max} = +/- 75 \text{ V}$$

$$V_o = +/- 50 \text{ V}$$

The gain of the A/D channel in PSIM will be set to 45.4545. The circuit connection and the settings are shown in the figure below.



Notice that in this circuit, the ac signal is sent to the A/D converter directly. This is because that, when the A/D input mode is set to AC, the input range is from -1.65V and +1.65V, and the function of the conditioning circuit that performs the dc offset is already included in the A/D converter block. In the actual hardware circuit, the ac signal must be scaled and offset so that the range is within 0V to +3.3V required by the A/D converter.

7.9 Comparator

F2803x support three comparator modules. Each comparator block can accommodate two external analog inputs, or use one external analog input and use the internal DAC reference for the other input. The comparator output can be sent to the PWM trip-zone and to the GPIO output.

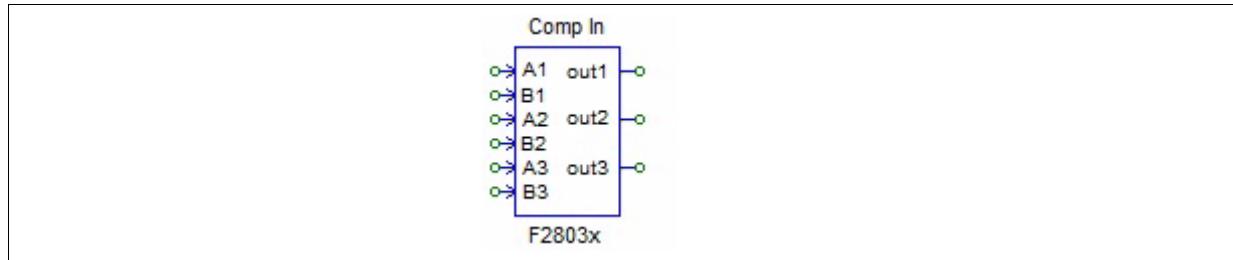
7.9.1 Comparator Input

In F2803x, there are 3 Comparators. The 3 pairs of inputs share the same ADC/AIO ports with ADC input channels as shown below:

- Comparator 1 input A - Port ADCA2/AIO2
- Comparator 1 input B - Port ADCB2/AIO10
- Comparator 2 input A - Port ADCA4/AIO4
- Comparator 2 input B - Port ADCB4/AIO12
- Comparator 3 input A - Port ADCA6/AIO6
- Comparator 3 input B - Port ADCB6/AIO14

Only one function can be designated for each port. Simcoder will report error if a port is defined as comparator input but is used as a A/D converter channel or AIO in the same PSIM circuit schematic.

Image:



Attributes

Parameters	Description
Comparator i	<p>Define how the output of the i_{th} comparator is used. It can be one of the following:</p> <ul style="list-style-type: none"> - <i>Do not use</i>: Not used - <i>As a normal comparator</i>: Used as a normal comparator - <i>As a Trip-Zone signal</i>: Used as a trip-zone signal
Output Logic	<p>Define the comparator output logic. It can be:</p> <ul style="list-style-type: none"> - <i>High when $A > B$</i>: The output is high when Input A is greater than B. - <i>High when $A < B$</i>: The output is high when Input A is less than B.
Compare Method	<p>Define how Input A of the comparator is compared to Input B. it can be one of the following:</p> <ul style="list-style-type: none"> - <i>Compare to Input B</i>: Input A is compared to Input B where Input B is an external analog signal. - <i>Compare to Constant Value</i>: Input A is compared to a constant value. - <i>Compare to DAC output</i>: Input A is compared to a DAC output which is an internal signal.
Constant Value	<p>The constant value when <i>Compare Method</i> is defined as <i>Compare to Constant Value</i>. The range of the constant value is from 0 to 3.3V.</p>

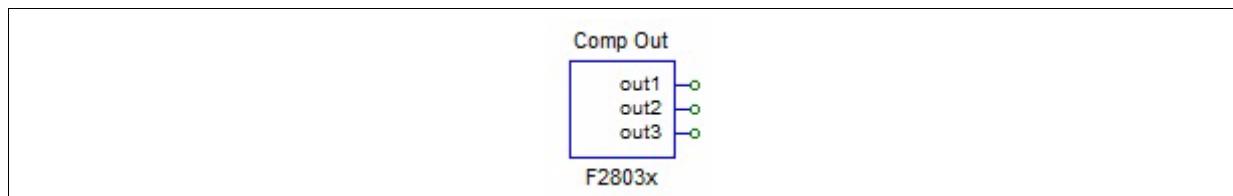
For all comparators, Input A is always from an external analog input. If Input B is also from another external analog input (if the parameter *Compare Method* is defined as *Compare to Input B*), the corresponding port must be defined as a comparator input in the Hardware Configuration block.

If the compare method is to compare to a constant value, Input B needs to be connected to ground in the schematic. If the compare method is to compare to a DAC output, Input B needs to be connected a comparator DAC output. In both cases, the corresponding ADC/AIO port can be used for other functions.

7.9.2 Comparator Output

The output of the comparator can be used as PWM trip-zone signal, as well as the GPIO output.

Image:



Attributes:

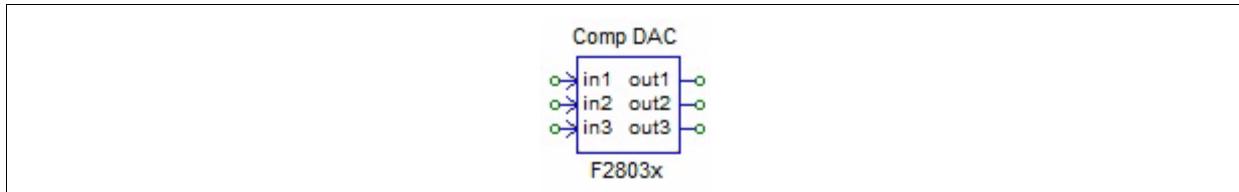
Parameters	Description
Comparator i Output	<p>Output port position of the i_{th} comparator. It can be:</p> <ul style="list-style-type: none"> - For Comparator A: GPIO1, 20, or 42 - For Comparator B: GPIO3, 21, 34, or 43 - For Comparator C: GPIO34

A comparator output block must be used together with a comparator input block. When a comparator output channel is used, the corresponding comparator input channel must be defined.

7.9.3 Comparator DAC

Each comparator block in F2803x contains a 10-bit DAC reference that can be used by the inverting input (Input B) of the comparator.

Image:



Attributes:

Parameters	Description
DAC i Range	The upper limit of the input signal range of the i_{th} comparator DAC. The lower limit is 0.
Use Ramp Generator	Define if the ramp generator is used in the comparator DAC.
Total Ramp Compensation	The total compensation of the ramp generator in one PWM period. It represents the total decrease of the ramp in one cycle.

Outputs of the comparator DAC can only be connected to the corresponding inverting inputs (Input B) of the comparator in a comparator input block. That is, node out1 can only be connected to node B1 of the comparator input block, and node out2 to node B2, and node out3 to B3. Also, a comparator DAC block cannot be used alone. It must be used in conjunction with a comparator input block.

If the ramp generator is not used, the input value is applied directly to DAC output immediately. The output range is from 0 to 3.3V, and the output can be calculated as follows:

$$\text{DAC Output} = \text{DAC Input} * 3.3 / \text{DAC Range}$$

If the ramp generator is used, the input value is saved and used as the initial output value in the next PWM period. The comparator DAC output decreases linearly within the PWM period, and the total decrease is equal to the total ramp compensation value.

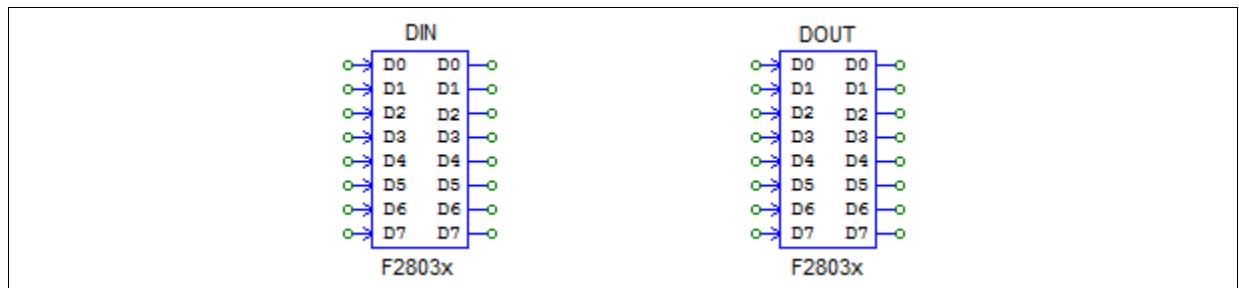
When the ramp generator is used, the sampling rate associated with the comparator DAC input must be the same as the frequency of the PWM generator that uses the comparator.

7.10 Digital Input and Digital Output

F2803x support 45 general-purpose-input-output (GPIO 0 to 44) ports that can be configured as either digital inputs or digital outputs. In addition, there are 6 digital analog-input-output ports (AIO 2, 4, 6, 10, 12, and 14) which also can be used as digital inputs and outputs.

In SimCoder, the digital inputs and outputs are grouped in 8-channel blocks. Multiple 8-channel digital input/output blocks can be used in the same schematic.

Images:



Attributes for Digital Input:

Parameters	Description
Port Position for Input i	The port position of the Input i , where i is from 0 to 7. It can be one of the 45 GPIO ports or one of the 6 AIO ports.
Use as External Interrupt	Indicate if this port is used as an external interrupt input.

Attributes for Digital Output:

Parameters	Description
Port Position for Output i	The port position of the Output i , where i is from 0 to 7. It can be one of the 45 GPIO ports or one of the 6 AIO ports.

Note that each GPIO and AIO port can be used for one function only. If an IO port is used as a digital input port, it can not be used as a digital output or any other peripheral port. For example, if Port GPIO1 is assigned as digital input and it is also used as PWM1 output, an error will be reported.

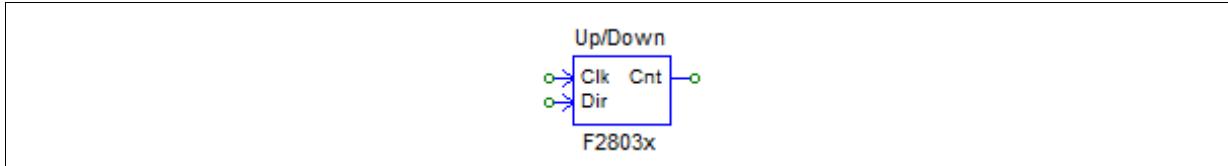
F2803x DSP supports 3 masked external interrupts (XINT1 to XINT3). There are no dedicated pins for the external interrupts. XINT1, XINT2, and XINT3 interrupts can accept inputs from GPIO0 to GPIO31 pins.

7.11 Up/Down Counter

F2803x supports one up/down counter. The input ports are:

- GPIO20 for clock
- GPIO21 for direction

Image:



Attributes:

In the image, "Clk" refers to the input clock signal, and "Dir" refers to the signal that defines the counting direction. When the "Dir" input is 1, the counter counts forward, and when the input is 0, the counter counts backward.

The output of the up/down counter gives the counter value.

Note that the up/down counter uses the same resource as the encoder, and the same GPIO ports cannot be used in a counter and in an encoder at the same time. For example, using both Encoder 1 and Up/Down Counter 1 will cause conflict and is not allowed.

7.12 Encoder and Encoder State

F2803x supports an **Encoder** module. The GPIO ports used by encoder are:

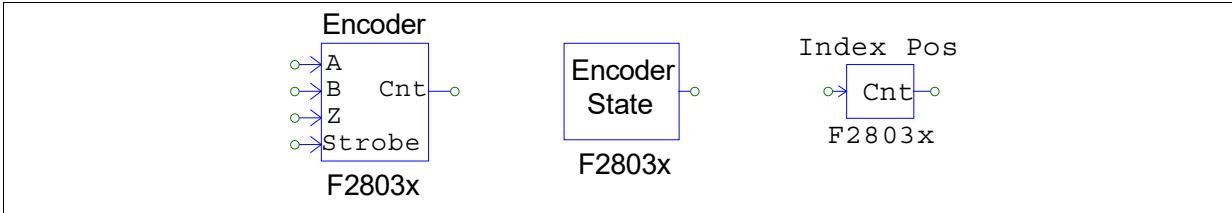
- GPIO 20 for clock input
- GPIO 21 for direction input
- GPIO 22 for Z (or index signal) input
- GPIO 23 for strobe signal input

The **Encoder State block** is used to indicate which input signal (either index signal or strobe signal) generates the interrupt. Also, hardware interrupt can be generated by the Z (index) signal and the strobe signal, and the output of the encoder state indicates which signal generates the interrupt. When the output is 0, the index signal generates the interrupt. When the output is 1, the strobe signal generates the interrupt.

The **Encoder Index/Strobe Position block** is used to latch the encoder's initial position. When the input of this

block is 0, the encoder counter is set to 0. Encoder will start to act when the input changes to 1. Encoder will latch the counter value when it meet the index/strobe event.

Images:



Attributes for Encoder:

Parameters	Description
Use Z Signal	Define if the encoder uses the Z (or index) signal.
Use Strobe Signal	Define if the encoder uses the strobe signal.
Counting Direction	The counting direction - Forward: the encoder counts up. - Reverse: the encoder counts down.
Z Signal Polarity	Define the trigger polarity of Z signal. - Active High - Active Low.
Strobe Signal Polarity	Define the trigger polarity of strobe signal. - Active High - Active Low.
Encoder Resolution	The resolution of the external encoder hardware. If it is 0, the encoder counter will keep on counting and will not reset. If for example, the resolution is set to 4096, the counter will be reset to 0 after it reaches 4095.

Attributes for Encoder Index/Strobe Pos:

Parameters	Description
Latch Position	Specify the kept counter type, choose from the followings: - IndexPos, if the setting "Use Z Signal" is not "No" in Encoder - StrobePos, if the setting "Use Strobe Signal" is not "No" in Encoder
Type of Position	This can be chosen from the followings: - The first latched position, or - The current latched position

7.13 Capture and Capture State

F2803x contains an enhanced capture module. A capture can generate interrupt, and the interrupt trigger mode is defined by the interrupt block.

Images:



Attributes for Capture:

Parameters	Description
Capture Source	Source of the capture may come from one of 3 GPIO ports, as listed below: - Capture 1 (GPIO 5) - Capture 1 (GPIO 19) - Capture 1 (GPIO 24)
Event Filter Prescale	Event filter prescale. The input signal is divided by the selected prescale.
Timer Mode	Capture counter timer mode. It can be either <i>Absolute time</i> or <i>Time difference</i> .

Attributes for Capture State:

Parameters	Description
Capture Source	Source of the capture. It has only one source <i>Capture1</i> .

The Capture State block output is either 1 or 0, where 1 means the rising edge and 0 means the falling edge.

7.14 Serial Communication Interface (SCI)

F2803x provides the function for serial communication interface (SCI). Through SCI, data inside the DSP can be transferred to a computer using an external RS-232 cable. PSIM provides all the necessary functions to transmit and receive data on both the DSP and computer sides, and to display the data on the computer. This provides a very convenient way to monitor, debug, and adjust the DSP code in real time.

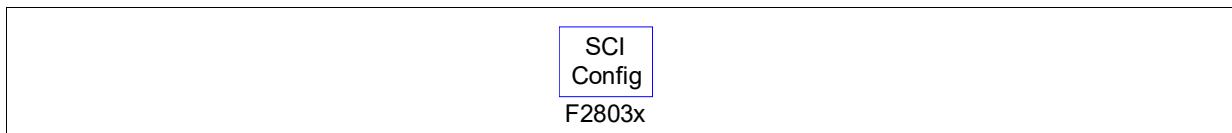
For more detailed descriptions on SCI and the monitoring function, please refer to the document "Tutorial - Using SCI for Real-Time Monitoring.pdf".

Three SCI function blocks are provided in SimCoder: *SCI Configuration*, *SCI Input*, and *SCI Output*.

7.14.1 SCI Configuration

The SCI Configuration block defines the SCI port, the communication speed, the parity check type, and the data buffer size.

Image:



Attributes:

Parameters	Description
SCI Port	Define the SCI port. Different sets of GPIO ports that can be used for SCI, as listed below: SCIA: GPIO 28 and 7 in combination with GPIO 29 and 12
Speed (bps)	SCI communication speed, in bps (bits per second). A list of preset speeds is provided at 200000, 115200, 57600, 38400, 19200, or 9600 bps. Or one can specify any other speed manually.
Parity Check	The parity check setting for error check in communication. It can be either <i>None</i> , <i>Odd</i> , or <i>Even</i> .
Output Buffer Size	Size of the data buffer allocated in DSP for SCI. The buffer is located in the RAM area, and each buffer cell stores one data point which consists of three 16-bit words (that is, 6 bytes, or 48 bits, per data point).

Note that the buffer size should be properly selected. On one hand, a large buffer is preferred in order to collect more data points so that more variables can be monitored over a longer period of time. On the other hand, the internal DSP memory is limited, and the buffer should not be too large to interfere with the normal DSP operation.

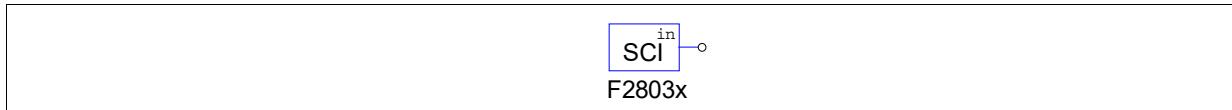
For more information on how to select the buffer size, please refer to the document "*Tutorial - Using SCI for Real-Time Monitoring.pdf*".

7.14.2 SCI Input

The SCI Input block is used to define a variable in the DSP code that can be changed. The name of the SCI input variable will appear in the DSP Oscilloscope (under the **Utilities** menu), and the value can be changed at runtime via SCI.

The SCI input block provides a convenient way to change reference, or fine tune controller parameters, for example.

Image:



Attributes:

Parameters	Description
Initial Value	The initial value of the SCI input variable.

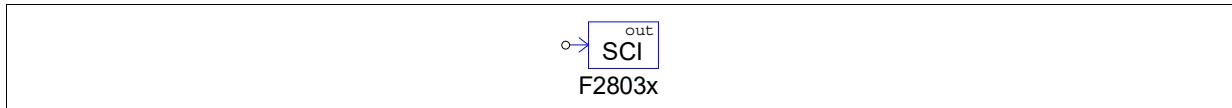
In the schematic, the SCI input behaves as a constant. While its value can be changed at runtime when the code is running on the DSP, the value will be fixed at the initial value in the simulation.

7.14.3 SCI Output

The SCI Output block is used to define a variable for display. When a SCI output block is connected to a node, the name of the SCI output block will appear in the DSP Oscilloscope (under the **Utilities** menu), and data of this variable can be transmitted from DSP to the computer via SCI at runtime, and the waveform can be displayed in the DSP Oscilloscope.

The SCI output block provides a convenient way to monitor DSP waveforms.

Image:



Attributes:

Parameters	Description
Data Point Step	It defines how frequent data is collected. If the Data Point Step is 1, every data point is collected and transmitted. If the Data Point Step is 10, for example, only one point of out every 10 points is collected and transmitted.

Note that if the Data Point Step is too small, there may be too many data points and it may not be possible to transmit them all. In this case, some data points will be discarded during the data transmission.

Also, the Data Point Step parameter is used only when the DSP Oscilloscope is in the *continuous* mode. When it is in the *snap-shot* node, this parameter is ignored and every point is collected and transmitted.

In simulation, the SCI output behaves as a voltage probe.

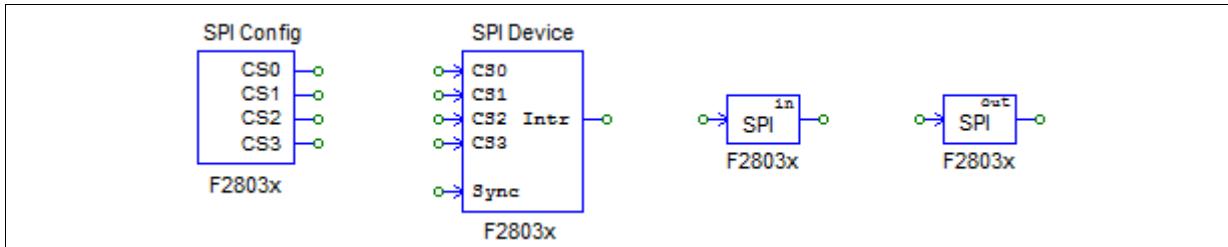
7.15 Serial Peripheral Interface (SPI)

F2803x provides the functions for serial peripheral interface (SPI). By using the SPI blocks in the TI F803x Target library, one can implement the function to communicate with external SPI devices (such as external A/D and D/A converters) easily and conveniently. Writing code manually for SPI devices is often a time-consuming and non-trivial task. With the capability to support SPI, PSIM greatly simplifies and speeds up the coding and hardware implementation process.

For more detailed descriptions on how to use SPI blocks, please refer to the document "*Tutorial - Using SPI for Real-Time Monitoring.pdf*".

Four SPI function blocks are provided in SimCoder: *SPI Configuration*, *SPI Device*, *SPI Input*, and *SPI Output*, as described below.

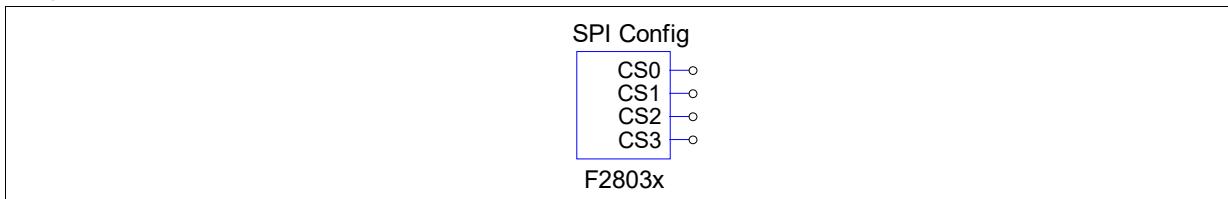
Images:



7.15.1 SPI Configuration

The SPI Configuration block defines the SPI port, the chip selection pins, and the SPI buffer size. It must be present in a schematic where SPI is used, and this block must be in the main schematic.

Image:



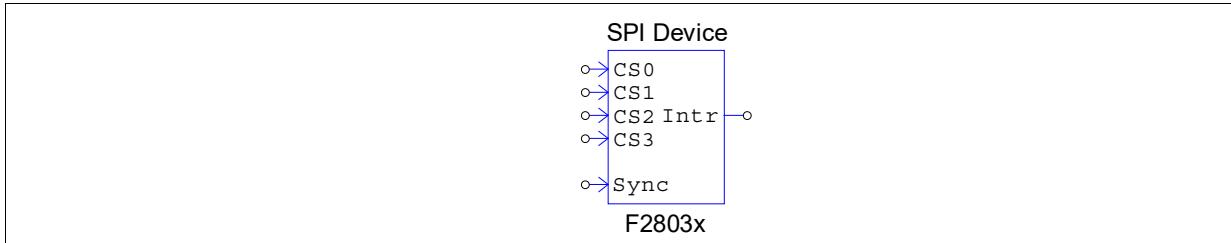
Attributes:

Parameters	Description
SPI Port	Define the SPI port from the options: <ul style="list-style-type: none">- SPIA (GPIO 16-19)- SPIA (GPIO 3, 5, 18, 19)- SPIB (GPIO 12-15)- SPIB (GPIO 24-27)
Chip Select Pin0, 1, 2, and 3	The GPIO port of the chip select pin. PSIM supports up to 16 SPI devices, which requires four GPIO pins for chip select, as defined by Chip Select Pin0 to Pin3. These GPIO ports and the SPI slave transmit-enable pin SPISTE are used to generate the chip select signal.
SPI Buffer Size	The buffer size of the SPI commands. Each memory cell of the buffer saves the index of a SPI command. Normally, one can specify the buffer size as 1 plus the number of SPI commands (i.e. Start Conversion Command, Receiving Data Command, Sending Data Command, and Sync. Command) in all SPI Input/Output elements.

7.15.2 SPI Device

The SPI Device block defines the information of the corresponding SPI hardware device. The number of SPI Device blocks in the schematic must be the same as the number of SPI hardware devices.

Image:



Attributes:

Parameters	Description
Chip Select Pins	The state of the chip select pins corresponding to the SPI device. When the chip select pins are at this state, this SPI device is selected.
Communication Speed (MHz)	SPI communication speed, in MHz.
Clock Type	SPI clock type, as determined by the SPI hardware device. It can be one of the following: - <i>Rising edge without delay</i> : The clock is normally low, and data is latched at the clock rising edge. - <i>Rising edge with delay</i> : The clock is normally low, and data is latched at the clock rising edge with delay. - <i>Falling edge without delay</i> : The clock is normally high, and data is latched at the clock falling edge. - <i>Falling edge with delay</i> : The clock is normally high, and data is latched at the clock falling edge with delay.
Command Word Length	Word length, or the length of the significant bits, of SPI communication commands. It can be from 1 to 16 bits.
Sync. Active Mode	The triggering mode of the synchronization signal of the SPI device. It can be either <i>Rising edge</i> or <i>Falling edge</i> .
SPI Initial Command	The SPI command that initializes the SPI device.
Hardware Interrupt Mode	Specify the type of the interrupt signal that the SPI device generates. This is valid only when the SPI device's interrupt output node is connected to the input of a digital output element. It can be one of the following: - <i>No hardware interrupt</i> - <i>Rising edge</i> - <i>Falling edge</i>

Interrupt Timing	<p>Specify how a SPI device generates interrupt when it completes conversion. It can be one of the following:</p> <ul style="list-style-type: none"> - <i>No interrupt</i>: No interrupt is generated. In this case, DSP sends the command to a SPI input device. This device starts the conversion and returns the result in the same command - <i>Multiple interrupt in series</i>: Multiple interrupts are generated in series after each conversion. This is for a SPI device that has one A/D conversion unit and multiple input channels. In this case, DSP send the first conversion command, and the SPI device starts the conversion. When the conversion is complete, the SPI device will generate an interrupt. In the interrupt service routine, DSP will send a command to fetch the conversion result, and start a new conversion of another channel of the same SPI input device. - <i>One-time interrupt</i>: Only one interrupt is generated at the end of the conversion. This is for a SPI device that can perform multiple channel conversions in one request. In this case, DSP sends the command to the SPI input device, and the SPI device completes the conversion of multiple input channels. When all the conversions are complete, the SPI device will generate an interrupt.
Command Gaps (ns)	The gap between two SPI commands, in nsec.
Conversion Sequence	Define the names of the SPI input elements, separated by comma, that determine the conversion sequence. Note that this parameter is valid only when the SPI device generates multiple interrupts in series.

In a schematic, the chip select pins of all the SPI devices are connected to the chip select pins of the SPI Configuration block, without defining how the chip select logic is implemented. In the actual hardware, however, one would need to implement the corresponding chip select logic accordingly.

A SPI command consists of a series of 16-bit numbers separated by comma. In the 16-bit number, only the lower bits are the significant bits used by the command. For example, if the Command Word Length is 8, Bits 0 to 7 are the command, and Bits 8 to 15 are not used.

A SPI device can be either an input device or an output device. For example, an external A/D converter is an input device. Usually DSP will send one or multiple A/D conversion commands to the device, and then set the synchronization signal to start the conversion. The synchronization signal is reset at the next command of the same device.

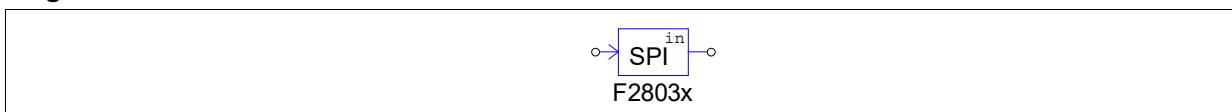
A SPI input device using the synchronization signal usually needs an interrupt pin to trigger DSP to enter the interrupt service routine.

On the other hand, an external D/A converter is an output device. Usually DSP sends one or multiple D/A conversion commands to the device, and then sets the synchronization signal to start the conversion. The synchronization signal is reset at the next command of the same device.

7.15.3 SPI Input

A SPI input device may have multiple input channels. The SPI Input block is used to define the properties of an input channel for SPI communication, and one SPI Input block corresponds to one input channel.

Image:



Attributes:

Parameters	Description
Device Name	Name of the SPI input device.
Start Conversion Command	Command to start conversion, in hex numbers, separated by comma (for example, 0x23,0x43,0x00).
Receiving Data Command	Command to receive data, in hex numbers, separated by comma (for example, 0x23,0x43,0x00).
Data Bit Position	Define where the data bits are in the receiving data string. The format is: $\text{ElementName} = \{Xn[\text{MSB..LSB}]\}$ <p>where</p> <ul style="list-style-type: none"> - ElementName is the name of the SPI input device. If it is the current SPI input device, use y instead. - $\{\}$ means that the item in the bracket repeats multiple times. - Xn is the n_{th} word received from the SPI input device, and n start from 0. - MSB..LSB defines the position of the significant bits in the word.
Input Range	Specify the parameter V_{max} that defines the input range. This parameter is valid only when the SPI device is an A/D converter. If the device conversion mode is DC, the input ranges from 0 to V_{max} . If the device conversion mode is AC, the input ranges from $-V_{max}/2$ to $V_{max}/2$.
Scale Factor	Output scale factor K_{scale} . If the scale factor is 0, the SPI device is not an A/D converter, and the result will be exactly the same as what DSP receives from SPI communication. Otherwise, the SPI device is an A/D converter, and the result is scaled based on this factor and the A/D conversion mode.
ADC Mode	The A/D conversion mode of the device. It can be either DC or AC. Note that this parameter is valid only when the device is an A/D converter.
Initial Value	The initial value of the input.

The formula for the *Data Bit Position* defines the data length of a SPI input device. For example, $y=x1[3..0]x2[7..0]$, means that the data length is 12, and the result is the lower 4 bits of the 2nd word and the lower 8 bits of the 3rd word. If the received data string is 0x12,0x78,0xAF, then the result is 0x8AF.

If the scale factor is not 0, the output will be scaled based on the following:

In the DC conversion mode:

$$\begin{aligned} \text{- In simulation: } & Output = Input \cdot K_{scale} \\ \text{- In hardware: } & Output = \frac{Result \cdot V_{max} \cdot K_{scale}}{2^{Data_Length}} \end{aligned}$$

In the AC conversion mode:

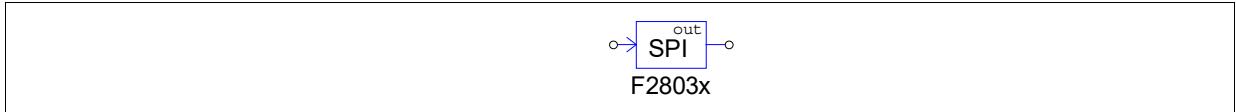
$$\begin{aligned} \text{- In simulation: } & Output = Input \cdot K_{scale} \\ \text{- In hardware: } & Output = \frac{(Result - 2^{Data_Length-1}) \cdot V_{max} \cdot K_{scale}}{2^{Data_Length-1}} \end{aligned}$$

The parameter *Data Length* is calculated from the Data Bit Position formula.

7.15.4 SPI Output

A SPI output device may have multiple output channels. The SPI Output block is used to define the properties of an output channel for SPI communication, and one SPI Output block corresponds to one output channel.

Image:



Attributes:

Parameters	Description
Device Name	Name of the SPI output device.
Scale Factor	Output scale factor K_{scale} . If the scale factor is 0, the SPI device is not a D/A converter, and the result will be exactly the same as what DSP receives from SPI communication. Otherwise, the SPI device is an D/A converter, and the result is scaled based on this factor and the D/A conversion mode.
Output Range	Specify the parameter V_{max} that defines the output range. This parameter is valid only when the SPI device is an D/A converter. If the device conversion mode is DC, the input ranges from 0 to V_{max} . If the device conversion mode is AC, the input ranges from $-V_{max}/2$ to $V_{max}/2$.
DAC Mode	The D/A conversion mode of the device. It can be either <i>DC</i> or <i>AC</i> . Note that this parameter is valid only when the device is a D/A converter.
Sending Data Command	Command to send the output data, in hex numbers, separated by comma (for example, 0x23,0x43,0x00).
Data Bit Position	Define where the data bits are in the sending data string. The format is: $ElementName = \{Xn[MSB..LSB]\}$ where <ul style="list-style-type: none"> - <i>ElementName</i> is the name of the SPI output device. If it is the current SPI output device, use <i>y</i> instead. - $\{\}$ means that the item in the bracket repeats multiple times. - <i>Xn</i> is the n_{th} word sent to the SPI output device, and <i>n</i> start from 0. - <i>MSB..LSB</i> defines the position of the significant bits in the word.
Sync. Command	The command to synchronize output channels of the SPI output device, in hex numbers, separated by comma (for example, 0x23,0x43,0x00). This command is used when the SPI output device does not have the synchronization signal

The formula for the *Data Bit Position* defines the data length of a SPI output device. For example, $y=x1[3..0]x2[7..0]$, means that the data length is 12, and the data is the lower 4 bits of the 2nd word and the lower 8 bits of the 3rd word. If the sending data string is 0x12,0x78,0xAF, then the data is 0x8AF.

If the scale factor is not 0, the output will be scaled based on the following:

In the DC conversion mode:

$$\begin{aligned} \text{- In simulation: } & Output = Input \cdot K_{scale} \\ \text{- In hardware: } & Output = \frac{Result \cdot K_{scale} \cdot 2^{Data_Length}}{V_{max}} \end{aligned}$$

In the AC conversion mode:

$$\begin{aligned} \text{- In simulation: } & Output = Input \cdot K_{scale} \\ \text{- In hardware: } & Output = 2^{Data_Length} + \frac{Result \cdot K_{scale} \cdot 2^{Data_Length-1}}{V_{max}} \end{aligned}$$

The parameter *Data_Length* is calculated from the Data Bit Position formula.

7.16 Controller Area Network (CAN) Bus

F2803x provides the function for CAN bus communication. PSIM provides the necessary functions to implement CAN bus.

Three function blocks are provided in SimCoder: *CAN Configuration*, *CAN Input*, and *CAN Output*, as described below.

7.16.1 CAN Configuration

The CAN Configuration block defines the CAN bus source, data byte order, and other settings.

F2803x DSP has one CAN source: CAN A with GPIO 31 for transmit and GPIO 30 for receive.

Image:



Attributes:

Parameters	Description
CAN Source	The CAN source can be one of the two groups: CAN A and CAN B. CAN A uses a combination of GPIO 19 and 31 for transmit, versus GPIO 18 and 30 for receive. CAN B uses a combination of GPIO 8, 12, 16 and 20 for transmit, versus GPIO 10, 13, 17, and 21 for receive.
CAN Speed	Communication speed, in Hz. It can be set to one of the following preset values: 125kHz, 250kHz, 500kHz, and 1MHz Or you can set the speed manually by typing the text in the parameter field. Note that the speed should not exceed 1MHz.
Data Byte Order	The order of the data bytes. It can be one of the following: - Least significant byte 1st - Most significant byte 1st "Least significant byte 1st" means that the least significant byte is placed first; while "Most significant byte 1st" means that the most significant byte is placed first.
Number of Input Mailboxes	Number of input mailboxes
Checking Receive Mail Lost	If it is set to <i>Enable</i> , the error report function " <code>_ProcCanAErrReport(nErr)</code> " (for CAN A) will be called if the received mail is lost. This function will return the value of the error status nErr. If it is set to <i>Disable</i> , the error report function will not be called.
Checking Bus Off	If it is set to <i>Enable</i> , the error report function " <code>_ProcCanAErrReport(nErr)</code> " (for CAN A) will be called if the bus is in the off state. This function will return the value of the error status nErr from the CAN register CANGIF0. If it is set to <i>Disable</i> , the error report function will not be called.
Error Check Mode	The error check mode can be either <i>Passive</i> or <i>Active</i> . If it is in the error-passive mode, an interrupt will be generated when the error count reaches 128. If it is in the error-active mode, an interrupt will be generated every time.

The returned status nErr in the function "`_ProcCanAErrReport(nErr)`" (for CAN A) is a 32-bit integer. It obtains

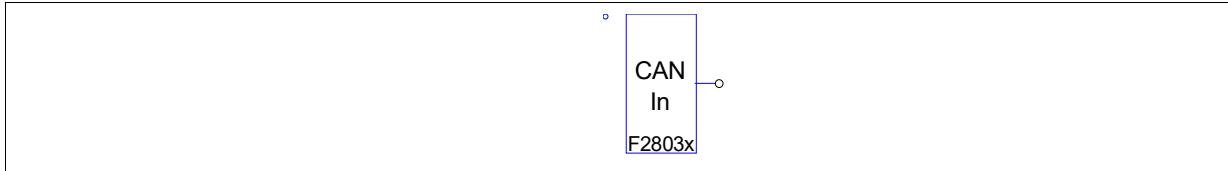
its value from the Global Interrupt Flag Register CANGIF0. After returning from the function "`_ProcCanAE rrReport(nErr)`", the register CANGIF0 will be cleared.

Also, if you wish to take actions on a specific error, you can add your own code within the "`_ProcCanAE rrReport(nErr)`" function.

7.16.2 CAN Input

A CAN Input block receives CAN messages from a CAN bus.

Image:



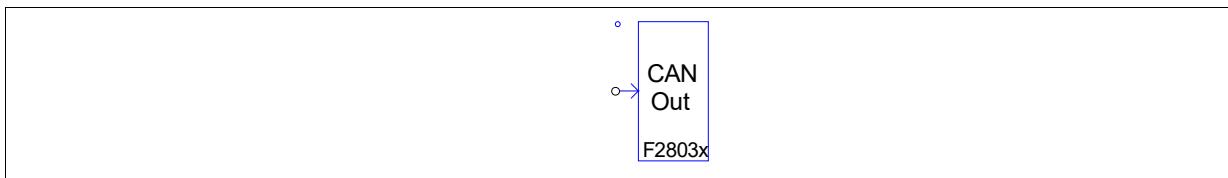
Attributes:

Parameters	Description
Number of Inputs	Number of CAN inputs. It can have up to 8 inputs.
Use Extension ID	If this is set to <i>Yes</i> , the ID of a message is a 29-bit integer. If this is set to <i>No</i> , the ID of a message is a 11-bit integer.
Message ID	The ID of a message. It is an integer, for example, 0x23.
Local Mask	The mask for the message. It is an integer. If the bits of the message ID matches the bits of the mask, the message will be received. Otherwise, it will be ignored. For example, if the mask is 0x380 and the message ID is 0x389, this message will be received. But if the message ID is 0x480, the message will be ignored.
Overwrite Flag	It can be set to <i>Allow overwrite</i> or <i>Do not allow overwrite</i> . Assume a mailbox is configured to accept a message, and there is a new message coming from the CAN bus, while the old message is still in the mailbox and has not been processed yet. If the flag is set to "Allow overwrite", the new message will be accepted, and the old message will be overwritten. If the flag is set to "Do not allow overwrite", the new message will not be accepted, and the old one will be kept.
Receive Message Rate	The number of messages received by the input block in a specific period of time. For example, there are two input blocks, with the first input block receiving 20 messages per second, and the second input block receiving 30 messages per second. The parameter "Receive Message Rate" will be set to 20 for the first block, and 30 for the second block.
Input <i>i</i> Gain	The gain to the <i>i</i> _{th} input where <i>i</i> can be 1 to 8. The output is the input multiplied by the gain.
Input <i>i</i> Data Start Position	A message can have up to 8 data points. A data point can have 1 bit up to 32 bits. This defines the start position of the current data point in the message.
Input <i>i</i> Data End Position	This defines the end position of the current data point in the message.
Input <i>i</i> Data Type	The data type can be either <i>Float</i> , <i>Integer</i> , or <i>IQ1</i> to <i>IQ30</i> .
Input <i>i</i> Default Data	The initial value of the SCI input variable.

7.16.3 CAN Output

A CAN Output block transmits CAN messages to a CAN bus.

Image:



Attributes:

Parameters	Description
Number of Outputs	Number of CAN outputs. It can have up to 8 outputs.
Use Extension ID	If this is set to <i>Yes</i> , the ID of a message is a 29-bit integer. If this is set to <i>No</i> , the ID of a message is a 11-bit integer.
Message ID	The ID of a message. It is an integer, for example, 0x23.
Trigger Type	The trigger type can be one of the following: - <i>No trigger</i> : No triggering - <i>Rising edge</i> : Triggering occurs at the rising edge of the trigger source. - <i>Falling edge</i> : Triggering occurs at the falling edge of the trigger source. - <i>Rising/falling edge</i> : Triggering occurs at both the rising edge and the falling edge of the trigger source. A rising/falling edge is considered to have occurred if the difference between the current value of the trigger source and the value at the last triggering instant is equal to or greater than 1.
Trigger Source	A trigger source can be either a constant or a global variable depending on the Trigger Type. If the Trigger Type is set to "No trigger", the trigger source defines the counter limit. For example, if the trigger source is a constant or a global value, and the value is 5, it means that triggering will occur once out of every 5 times. In another word, the data will be sent out once per every 5 cycles. If the Trigger Type is set to edge trigger, the trigger source can only be a global variable. Triggering will occur when the global variable has the rising or falling edge or both depending on the Trigger Type.
Output _i Gain	The gain to the i_{th} output where i can be 1 to 8. The output is the output multiplied by the gain.
Output _i Data Start Position	A message can have up to 8 data points. A data point can have 1 bit up to 32 bits. This defines the start position of the current data point in the message.
Output _i Data End Position	This defines the end position of the current data point in the message.
Output _i Data Type	The data type can be either <i>Float</i> , <i>Integer</i> , or <i>IQ1</i> to <i>IQ30</i> .
Output _i Default Data	The initial value of the SCI output variable.

7.17 Interrupt Time

The interrupt time block is used to measure the time interval of an interrupt service routine.

Attributes:

Parameters	Description
Time Output Method	Define how interrupt time is measured. It can be one of the following: - <i>SCI (time used)</i> : Using SCI. Time used by the interrupt service routine, in DSP clock count, is measured and is sent out via SCI output. - <i>SCI (time remaining)</i> : Using SCI. Time remaining in the interrupt service routine, in DSP clock count, is measured and is send out via SCI output. The time remaining is defined as the time from the end of the current interrupt to the beginning of the next interrupt. - <i>GPIO0 to GPIO44 or AIO2 to AIO14</i> : Using a GPIO port. A pulse is generated at the specified GPIO port. The pulse is set to high when entering the interrupt, and set to low when exiting the interrupt. An oscilloscope can be used to measure the width of the pulse.
Sampling Frequency	Sampling frequency of the interrupt service routine, in Hz.

When SCI is used, the value is the count of the DSP clock. For example, if the value is 6000, for a 60-MHz DSP clock, the interrupt time will be: $6000 / 60M = 100 \text{ us}$.

7.18 Project Settings and Memory Allocation

When generating the code for the TI F2803x Hardware Target, SimCoder also creates the complete project files for the TI Code Composer Studio (CCS) development environment, so that the code can be compiled, linked, and uploaded to DSP.

At the present, CCS version 3.3 is supported. Assuming that the PSIM schematic file is "test.sch", after the code generation, a sub-folder called "test (C code)" will be generated in the directory of the schematic file, and sub-folder will contain the following files:

- test.c Generated C code
- PS_bios.h: Header file for the SimCoder F2803x library
- passwords.asm: File for specifying the DSP code password
- test.pjt: Project file for Code Composer Studio
- F2803x_Headers_nonBIOS.cmd: Peripheral register linker command file
- F28035_FLASH_Lnk.cmd: Flash memory linker command file
- F28035_FLASH_RAM_Lnk.cmd: Flash RAM memory linker command file
- F28035_RAM_Lnk.cmd: RAM memory linker command file

Note: The names of the link command files are assigned with the target hardware if it is not F28035. For example, if the target hardware is F28034, the file names will be *F28034 FLASH Lnk.cmd*, *F28034 FLASH RAM Lnk.cmd*, and *F28034RAM Lnk.cmd* accordingly.

Besides, the project also needs the following library files:

- PsBiosRamF03xFixpt.lib: SimCoder F2803x library, located in the PSIM folder
- PsBiosRomF03xFixpt.lib: SimCoder F2803x library, located in the PSIM folder
- IQmath.lib: TI's IQmath.lib, located in the PSIM /lib folder
- 2803x_IQmath_BootROMsymbols.libIQmath symbols library, located in the PSIM /lib folder

These library files will be copied automatically to the project folder when the code is generated.

Each time code generation is performed, the .c file and .pjt file (in this example, "test.c" and "test.pjt") will be created. If you have made changes manually to these two files, be sure to copy the changed files to a different location. Otherwise the changes will be overwritten when code generation is performed next time.

Project Setting:

In the Code Composer Studio project file, the following settings are provided:

- *RAM Debug*: To compile the code in the debug mode and run it in the RAM memory
- *RAM Release*: To compile the code in the release mode and run it in the RAM memory
- *Flash Release*: To compile the code in the release mode and run it in the flash memory
- *Flash RAM Release*: To compile the code in the release mode and run it in the RAM memory

When RAM Debug or RAM Release is selected, CCS uses the linker command file F2803x_RAM_Lnk.cmd to allocate the program and data space.

When Flash Release is selected, CCS uses the linker command file F2803x_FLASH_Lnk.cmd to allocate the program and data space.

When Flash RAM Release is selected, CCS uses the linker command file F2803x_FLASH_RAM_Lnk.cmd to allocate the program and data space. The memory allocation is the same as in RAM Release.

The code compiled in the release mode is faster than the code in the debug mode. Also, the code in RAM Release or Flash RAM Release is the fastest. The code in RAM Debug is slower, and the code in Flash Release is the slowest. In a development, normally one would start with RAM Debug for easy debugging. Then switch to RAM Release and consequently to Flash Release or Flash RAM Release.

Memory Allocation:

In the generated link files, the memory allocation is defined in the following way.

With the RAM Debug, RAM Release, and Flash RAM Release settings:

RAM Memory 0x0000 - 0x07FF (2K) interrupt vectors stack 0x8000 - 0x9FFF (8K*) program and data space
--

With the Flash Release setting:

RAM Memory 0x0000 - 0x07FF (2K) interrupt vectors stack 0x8000 - 0x9FFF (8K*) data space	Flash Memory 0x3E8000 - 0x3F7FFF (64K**) program password etc.
--	---

Notes:

* The RAM memory predefined by SimCoder for program and data space is:

- For F28035, F28034, F28032, and F28032: from 0x8000 to 0x9FFF (8K)
- For F28031: from 0x8000 to 0x97FF (6K)
- For F28030: from 0x8000 to 0x8FFF (4K)
- If the combined program and data space exceeds the size of the RAM space, Flash Release must be selected as the project setting.

** The flash memory predefined by SimCoder for program space is:

- For F28035 and F28034: from 0x3E8000 to 0x3F7FFF (64K)
- For F28033, F28032, and F28031: from 0x3F0000 to 0x3F7FFF (32K)
- For F28030: from 0x3F4000 to 0x3F7FFF (16K)

F2806x Hardware Target

8.1 Overview

With the F2806x Hardware Target, SimCoder can generate code that is ready to run on any hardware boards based on Texas Instruments' F2806x fixed-point DSP.

The F2806x Hardware Target will work with all F2806x packages.

The F2806x Hardware Target library includes the following function blocks:

- PWM generators: 3-phase, 2-phase, 1-phase, and APWM
- Variable frequency PWM
- Start/Stop functions for PWM generators
- Trip-one and trip-zone state
- A/D converter
- Comparator input, output, and DAC
- Digital input and output
- Up/Down counter
- Encoder and encoder state
- Capture and capture state
- SCI configuration, Input, and output
- SPI configuration, device, input, and output
- CAN configuration, input, and output
- Interrupt Time
- DSP clock
- Hardware configuration

When generating the code for a system that has multiple sampling rates, SimCoder will use the interrupts of the PWM generators for the PWM sampling rates. For other sampling rates in the control system, it will use the Timer 1 interrupt first, and then Timer 2 interrupt if needed. If there are more than three sampling rates in the control system, the corresponding interrupt routines will be handled in the main program by software.

In TI F2806x, PWM generators can generate hardware interrupt. SimCoder will search and group all the elements that are connected to the PWM generator and have the same sampling rate as the PWM generator. These elements will be automatically placed and implemented in an interrupt service routine in the generated code.

In addition, digital input, encoder, capture, and trip-zone can also generate hardware interrupt. Each hardware interrupt must be associated with an interrupt block (described in Section 4.5 of this Manual), and each interrupt block must be associated with an interrupt service routine (a subcircuit that represents the interrupt service routine). For example, if a PWM generator and a digital input both generate interrupt, there should be one interrupt block and one interrupt service routine for each of them.

The definitions of the elements in the F2806x Hardware Target library are described in this Chapter.

The figure below shows the F2806x 100-pin PZ port assignment.

F2806x 100-Pin PZ Port Assignment

GPIO42/EPWM8A/TZ1/COMP1OUT		1	50	GPIO28/SCIRXDA/SDAA/TZ2
GPIO23/EQEP1I/MFSXA/SCIRXDB		2	49	GPIO9/EPWM5B/SCITXDB/ECAP3
V _{DD}		3	48	GPIO51/EQEP1B/MDRA/TZ2
V _{SS}		4	47	V _{SS}
V _{DDIO}		5	46	V _{DD3VFL}
GPIO20/EQEP1A/MDXA/COMP1OUT		6	45	TEST2
GPIO21/EQEP1B/MDRA/COMP2OUT		7	44	GPIO12/TZ1/SCITXDA/SPISIMOB
GPIO43/EPWM8B/TZ2/COMP2OUT		8	43	GPIO29/SCITXDA/SCLA/TZ3
GPIO4/EPWM3A		9	42	GPIO50/EQEP1A/MDXA/TZ1
GPIO5/EPWM3B/SPISIMO/A/ECAP1		10	41	GPIO30/CANRXA/EQEP2I/EPWM7A
XRS		11	40	GPIO31/CANTXA/EQEP2S/EPWM8A
TRST		12	39	GPIO25/ECAP2/EQEP2B/SPISOMIB
V _{DDIO}		13	38	V _{DDIO}
V _{DD}		14	37	V _{DD}
V _{SS}		15	36	V _{SS}
ADCINA7		16	35	ADCINB7
ADCINA6/COMP3A/AI06		17	34	ADCINB6/COMP3B/AI014
ADCINA5		18	33	ADCINB5
ADCINA4/COMP2A/AI04		19	32	ADCINB4/COMP2B/AI012
ADCINA3		20	31	ADCINB3
ADCINA2/COMP1A/AI02		21	30	ADCINB2/COMP1B/AI010
ADCINA1		22	29	ADCINB1
ADCINA0		23	28	ADCINB0
V _{REFHI}		24	27	V _{REFLO}
V _{DDA}		25	26	V _{SSA}
GPIO41/EPWM7B/SCIRXDB		76	75	GPIO55/SPISIMO/A/EQEP2B/HRCAP2
GPIO27/HRCAP2/EQEP2S/SPISTEB/USB0DM		77	74	GPIO10/EPWM6A/ADCSOCBO
GPIO26/ECAP3/EQEP2I/SPICLKB/USB0DP		78	73	GPIO11/EPWM6B/SCIRXDB/ECAP1
V _{DDIO}		79	72	GPIO36/TMS
V _{SS}		80	71	GPIO35/TDI
V _{DD}		81	70	GPIO37/TDO
GPIO40/EPWM7A/SCITXDB		82	69	GPIO54/SPISIMO/A/EQEP2A/HRCAP1
GPIO3/EPWM2B/SPISIMO/A/COMP2OUT		83	68	GPIO34/COMP2OUT/COMP3OUT
GPIO2/EPWM2A		84	67	GPIO38/XCLKIN/TCK
GPIO56/SPICLKA/EQEP2I/HRCAP3		85	66	GPIO39
GPIO1/EPWM1B/COMP1OUT		86	65	GPIO53/EQEP1I/MFSXA
GPIO0/EPWM1A		87	64	GPIO19/XCLKIN/SPISTEĀ/SCIRXDB/ECAP1
GPIO15/ECAP2/SCIRXDB/SPISTEB		88	63	V _{DD}
GPIO57/SPISTEĀ/EQEP2S/HRCAP4		89	62	V _{SS}
V _{REGENZ}		90	61	V _{DDIO}
V _{DD}		91	60	X1
V _{SS}		92	59	X2
V _{DDIO}		93	58	GPIO6/EPWM4A/EPWMSYNCI/EPWMSYNCO
GPIO58/MCLKRA/SCITXDB/EPWM7A		94	57	GPIO7/EPWM4B/SCIRXDA/ECAP2
GPIO13/TZ2/SPISOMIB		95	56	GPIO44/MFSRA/SCIRXDB/EPWM7B
GPIO14/TZ3/SCITXDB/SPICLKB		96	55	GPIO16/SPISIMO/A/TZ2
GPIO24/ECAP1/EQEP2A/SPISIMO/B		97	54	GPIO8/EPWM5A/ADCSOCAO
GPIO22/EQEP1S/MCLKXA/SCITXDB		98	53	GPIO52/EQEP1S/MCLKXA/TZ3
GPIO32/SDAA/EPWMSYNCI/ADCSOCAO		99	52	GPIO17/SPISIMO/A/TZ3
GPIO33/SCLA/EPWMSYNCO/ADCSOCBO		100	51	GPIO18/SPICLKA/SCITXDB/XCLKOUT

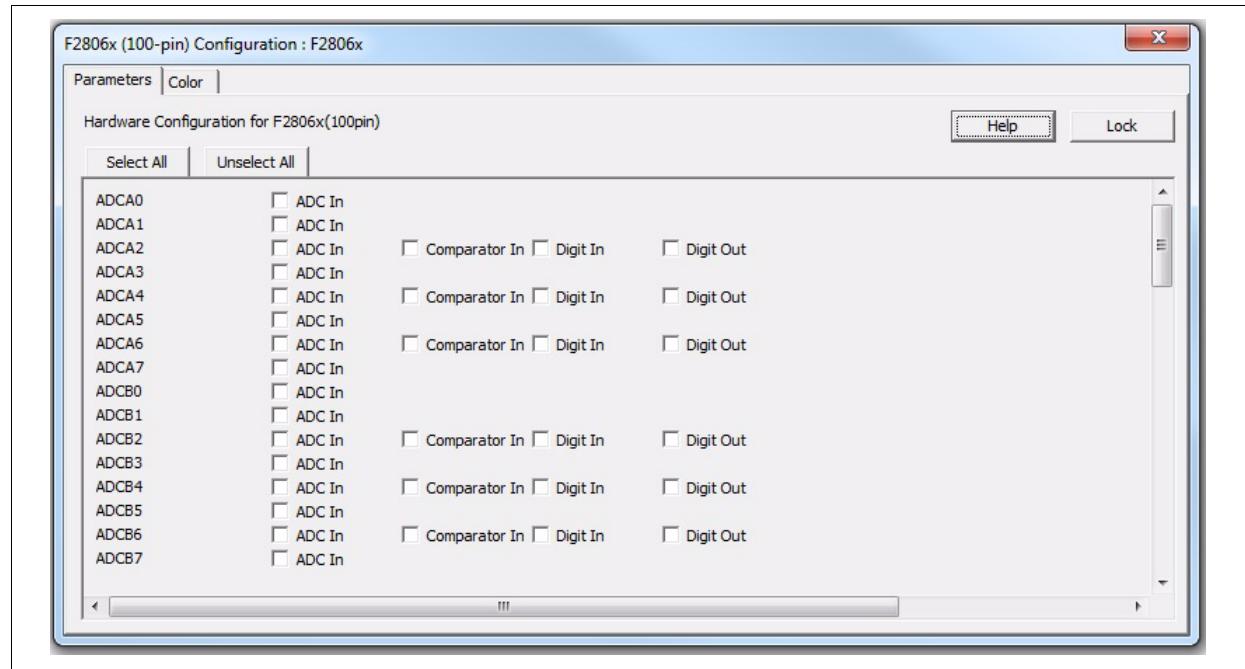
8.2 Hardware Configuration

F2806x provides a 16 channel analog inputs and up to 45 individually programmable multiplexed GPIO ports. Many of the GPIO ports can perform one of several functions. For example, port GPIO1 can be used either as a digital input, a digital output, or a PWM output. Some of the 16 A/D channels can also be defined as either digital input, digital output, or comparator input. Therefore, user must assign the functions to the GPIO ports correctly according to the PSIM circuit schematic.

Image:



The dialog window of the block is shown below:



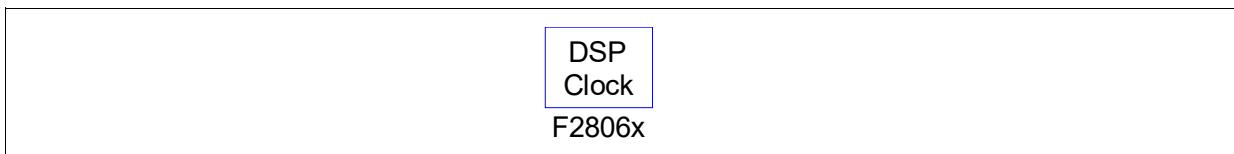
The Hardware Configuration block is for user to specify the I/O ports of the F2806x hardware. Every port to be used must be assigned correctly. The ports not in use can be left unchecked.

For each GPIO port, a check box is provided for each of its available function. When a box is checked, the GPIO port is configured for that particular function. For example, if the checkbox for "Digital Input" is checked for port GPIO1, this port is configured as a digital input, and hence, cannot be used for any other functions. If it is used as a PWM output in the PSIM circuit schematic, an error message will be generated.

8.3 DSP Clock

The DSP Configuration block defines the external clock frequency and the speed of the F2806x DSP, as well as the program space size.

Image:



Attributes:

Parameters	Description
DSP Clock Source	There are five ways of providing system clock to F2806x. They are as follows: <ul style="list-style-type: none">- Internal oscillator 1- Internal oscillator 2- External oscillator- External clock (GPIO19)- External clock (GPIO38)
External Clock (MHz)	Frequency of the external clock on the DSP board, in MHz. The frequency must be an integer, and the maximum frequency allowed is 10 MHz. This parameter is ignored if the DSP clock source is selected to be internal oscillator 1 or 2.
DSP Speed (MHz)	DSP Speed, in MHz. The speed must be an integer, and must be an integer multiple of the external clock frequency, from 1 to 12 times. The maximum DSP speed allowed is 90 MHz.

If a DSP Configuration block is not used in a circuit, the default values of the DSP Configuration block will be used.

8.4 PWM Generators

F2806x contains 8 sets of PWM modules. Each set of PWM module has two output ports:

- PWM 1: GPIO 0 and 1
- PWM 2: GPIO 2 and 3
- PWM 3: GPIO 4 and 5
- PWM 4: GPIO 6 and 7
- PWM 5: GPIO 8 and 9
- PWM 6: GPIO 10 and 11
- PWM 7: GPIO 30, 40, and 58, in pair with GPIO 41 and 44
- PWM 8: GPIO 31 and 42, in pair with GPIO 43

The two outputs of each PWM module usually are complementary to each other. For example PWM 1 has a positive output PWM 1A and a negative output PWM 1B, except when the PWM module is set in one of a few special operation modes.

In SimCoder, these 8 PWM's can be used in the following ways:

- Two 3-phase PWM generators: PWM 123 (consisting of PWM 1, 2, and 3) and PWM 456 (consisting of PWM 4, 5, and 6);
- Seven 2-phase PWM generators: PWM 1, 2, 3, 4, 5, 6, 7, and 8, with the two outputs of each PWM generator not in a complementary way, but in special operation mode.
- 1-phase PWM generators: PWM 1, 2, 3, 4, 5, 6, 7, and 8, with two outputs complementary to each other.
- 1-phase PWM generators with phase shift: PWM 2, 3, 4, 5, 6, 7, and 8, with two outputs complementary to each other.

These PWM generators can trigger the A/D converter, and use trip-zone signals.

Beside the PWM generators described above, there are also 6 APWM generators that use the same resources as the captures. These PWM generators have restricted functionality as compared to the 6 PWM generators (PWM 1 to 6) as they can not trigger the A/D converter and can not use trip-zone signals. Also, because of the common resources, when a particular port is used for the capture, it can not be used for the PWM generator.

Note that all the PWM generators in SimCoder include one switching period delay internally. That is, the input value of a PWM generator is delayed by one cycle before it is used to update the PWM output. This delay is needed to simulate the delay inherent in the DSP hardware implementation.

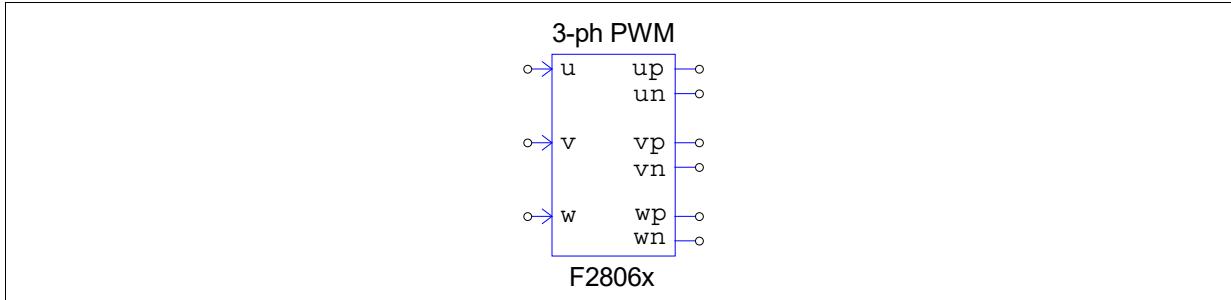
PWM generators have a parameter called "PWM Freq. Scaling Factor". It can be set to 1 to 100. The hardware

limit is 3. If the scaling factor is greater than 3, PWM will use an unused PWM to generate interrupt at the sampling frequency. This unused PWM is only used to generate a periodic interrupt, and its outputs can still be used for other functions. If there is no unused PWM in the system, a timer will be used.

8.4.1 3-Phase PWM

In the 3-phase PWM generator image, "u", "v", and "w" refer to the three phases (alternatively they are called Phase "a", "b", and "c"). The letter "p" refers to the positive output, and "n" refers to the negative output. For example, for 3-phase PWM 123, "up" is PWM1A, and "un" is PWM1B.

Image:



Attributes:

Parameters	Description
PWM Source	Source of the PWM generator. It can be either "3-phase PWM 123" that uses PWM 1 to 3, or "3-phase PWM 456" that uses PWM 4 to 6.
Dead Time	The dead time T_d for the PWM generator, in sec.
Sampling Frequency	Sampling frequency of the PWM generator, in Hz. The calculation is done and the PWM signal duty cycle is updated based on this frequency.
PWM Freq. Scaling Factor	The ratio between the PWM switching frequency and the sampling frequency. It can be from 1 to 100. For example, if the sampling frequency is 50 kHz and the scaling factor is 3, the PWM switching frequency will be 150 kHz. That is switches will operate at 150 kHz. But gating signals will be updated at 50 kHz, or once per 3 switching cycles.
Carrier Wave Type	Carrier wave type and the initial PWM output state. It can be one of the following: <ul style="list-style-type: none"> - <i>Triangular (start low)</i>: Triangular wave, and the initial PWM output state is low. - <i>Triangular (start high)</i>: Triangular wave, and the initial output state is high. - <i>Sawtooth (start low)</i>: Sawtooth wave, and the initial output state is low. - <i>Sawtooth (start high)</i>: Sawtooth wave, with the initial output state is high.
Trigger ADC	Setting whether for the PWM generator to trigger the A/D converter. It can be one of the following: <ul style="list-style-type: none"> - <i>Do not trigger ADC</i>: PWM does not trigger the A/D converter. - <i>Trigger ADC</i>: PWM will trigger A/D converter.
ADC Trigger Position	A/D trigger position ranges from 0 to a value less than 1. When it is 0, the A/D converter is triggered at the beginning of the PWM cycle, and when it is 0.5, the A/D converter is triggered at the 180° position of the PWM cycle.

Use Trip-Zone i PWMA DC Trip Src1(DCAH) PWMA DC Trip Src1(DCAL) PWMA 1-shot Evt (DCAEVT1) PWMA CBC Evt (DCAEVT1)	<p>Define whether the PWM generator uses the i_{th} trip-zone signal or not, where i ranges from 1 to 6. It can be one of the following:</p> <ul style="list-style-type: none"> - <i>Disable Trip-Zone i</i>: Disable the i_{th} trip-zone signal. - <i>One shot</i>: The PWM generator uses the trip-zone signal in the one-shot mode. Once triggered, the PWM must be started manually. - <i>Cycle by cycle</i>: The PWM generator uses the trip-zone signal in the cycle-by-cycle basis. The trip-zone signal is effective within the current cycle, and PWM will automatically re-start in the next cycle. <p>Digital compare (DC) trip source DCAH for PWMA. For a 3-phase PWM generator, PWMA refers to the outputs "up", "up", and "up" for the 3 top switches. The PWM channel may have up to two DC trip sources: DCAH and DCAL. The trip source can be one of the following:</p> <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>Trip-zone 1</i>: PWM uses trip-zone 1 as digital compare input signal. - <i>Trip-zone 2</i>: PWM uses trip-zone 2 as digital compare input signal. - <i>Trip-zone 3</i>: PWM uses trip-zone 3 as digital compare input signal. - <i>Comparator 1</i>: PWM uses Comparator 1 as digital compare input signal. - <i>Comparator 2</i>: PWM uses Comparator 2 as digital compare input signal. - <i>Comparator 3</i>: PWM uses Comparator 3 as digital compare input signal. <p>Digital compare (DC) trip source DCAL for PWMA. The trip source can be one of the following:</p> <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>Trip-zone 1</i>: PWM uses trip-zone 1 as digital compare input signal. - <i>Trip-zone 2</i>: PWM uses trip-zone 2 as digital compare input signal. - <i>Trip-zone 3</i>: PWM uses trip-zone 3 as digital compare input signal. - <i>Comparator 1</i>: PWM uses Comparator 1 as digital compare input signal. - <i>Comparator 2</i>: PWM uses Comparator 2 as digital compare input signal. - <i>Comparator 3</i>: PWM uses Comparator 3 as digital compare input signal. <p>Define how the one-shot signal is used for the DC trip signal of PWMA. The active level of the DC trip signal can be selected from the following:</p> <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>Trip source 1 is low</i>: PWM is tripped if the source signal 1 is low. - <i>Trip source 1 is high</i>: PWM is tripped if the source signal 1 is high. - <i>Trip source 2 is low</i>: PWM is tripped if the source signal 2 is low. - <i>Trip source 3 is high</i>: PWM is tripped if the source signal 2 is high. - <i>Trip source 1 low & source 2 high</i>: PWM is tripped if the source 1 signal is low and at the same time source 2 signal is high. <p>Define how the cycle-by-cycle signal is used for the DC trip signal of PWMA. The active level of the DC trip signal can be selected from the following:</p> <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>Trip source 1 is low</i>: PWM is tripped if the source signal 1 is low. - <i>Trip source 1 is high</i>: PWM is tripped if the source signal 1 is high. - <i>Trip source 2 is low</i>: PWM is tripped if the source signal 2 is low. - <i>Trip source 3 is high</i>: PWM is tripped if the source signal 2 is high. - <i>Trip source 1 low & source 2 high</i>: PWM is tripped if the source 1 signal is low and at the same time source 2 signal is high.
--	---

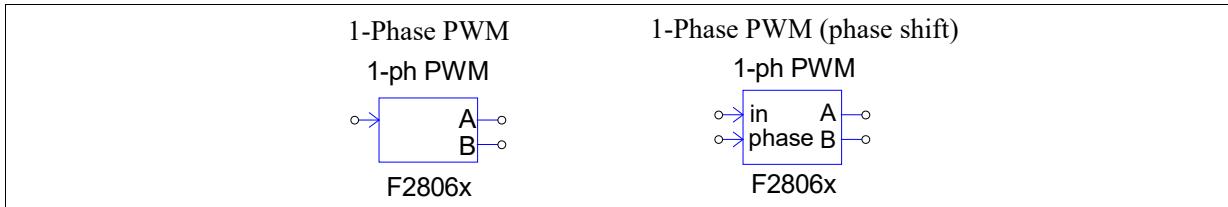
PWMB DC Trip Src1(DCBH)	Digital compare (DC) trip source DCBH for PWMB. For a 3-phase PWM generator, PWMB refers to the outputs "un", "vn", and "wn" for the 3 bottom switches. The PWM channel may have up to two DC trip sources: DCBH and DCBL. The trip source can be one of the following: <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>Trip-zone 1</i>: PWM uses trip-zone 1 as digital compare input signal. - <i>Trip-zone 2</i>: PWM uses trip-zone 2 as digital compare input signal. - <i>Trip-zone 3</i>: PWM uses trip-zone 3 as digital compare input signal. - <i>Comparator 1</i>: PWM uses Comparator 1 as digital compare input signal. - <i>Comparator 2</i>: PWM uses Comparator 2 as digital compare input signal. - <i>Comparator 3</i>: PWM uses Comparator 3 as digital compare input signal.
PWMB DC Trip Src1(DCBL)	Digital compare (DC) trip source DCBL for PWMB. The trip source can be one of the following: <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>Trip-zone 1</i>: PWM uses trip-zone 1 as digital compare input signal. - <i>Trip-zone 2</i>: PWM uses trip-zone 2 as digital compare input signal. - <i>Trip-zone 3</i>: PWM uses trip-zone 3 as digital compare input signal. - <i>Comparator 1</i>: PWM uses Comparator 1 as digital compare input signal. - <i>Comparator 2</i>: PWM uses Comparator 2 as digital compare input signal. - <i>Comparator 3</i>: PWM uses Comparator 3 as digital compare input signal.
PWMB 1-shot Evt (DCBEVT1)	Define how the one-shot signal is used for the DC trip signal of PWMB. The active level of the DC trip signal can be selected from the following: <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>Trip source 1 is low</i>: PWM is tripped if the source signal 1 is low. - <i>Trip source 1 is high</i>: PWM is tripped if the source signal 1 is high. - <i>Trip source 2 is low</i>: PWM is tripped if the source signal 2 is low. - <i>Trip source 3 is high</i>: PWM is tripped if the source signal 2 is high. - <i>Trip source 1 low & source 2 high</i>: PWM is tripped if the source 1 signal is low and at the same time source 2 signal is high.
PWMB CBC Evt (DCBEVT1)	Define how the cycle-by-cycle signal is used for the DC trip signal of PWMB. The active level of the DC trip signal can be selected from the following: <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>Trip source 1 is low</i>: PWM is tripped if the source signal 1 is low. - <i>Trip source 1 is high</i>: PWM is tripped if the source signal 1 is high. - <i>Trip source 2 is low</i>: PWM is tripped if the source signal 2 is low. - <i>Trip source 3 is high</i>: PWM is tripped if the source signal 2 is high. - <i>Trip source 1 low & source 2 high</i>: PWM is tripped if the source 1 signal is low and at the same time source 2 signal is high.
DC Event Filter Source	Source of the digital compare event filter. It can be one of the following: <ul style="list-style-type: none"> - <i>Do not use</i>: Do not use DC event filter. - <i>DCAEVT1</i>: Use DCAEVT1 as the filter source. - <i>DCAEVT2</i>: Use DCAEVT2 as the filter source. - <i>DCBEVT1</i>: Use DCBEVT1 as the filter source. - <i>DCBEVT2</i>: Use DCBEVT2 as the filter source. - <i>DCAEVT2</i>: PWM is tripped if the source signal 1 is high.
Blanking Window Pos (us)	Blanking window start position in a PWM period, in us.
Blanking Window Width (us)	Width of the blanking window, in us. The width is limited by the hardware, and can be calculated as below: $\text{255} / \text{CPU Frequency}$ <p>For example, when the CPU speed is 90MHz, the width range is 0 to 2.83us.</p>

Blanking Window Range	Specify how the blanking window is applied. It can be one of the following: <ul style="list-style-type: none"> - <i>In the window</i>: The blanking action is applied with the window defined (from the start position for a window width defined) - <i>Out of window</i>: The blanking action is applied outside the window defined.
Applying Event Filtering	Specify how event filtering is applied to digital compare events. The event filtering can be applied to any combinations of the following digital compare events: DCAEVT1, DCAEVT2, DCBEVT1, and DCBEVT2
Trip Action	Define how the PWM generator responds to the trip action. It can be one of the following: <ul style="list-style-type: none"> - <i>High impedance</i>: PWM outputs in high impedance - <i>PWM A high & B low</i>: Set PWM A high and B low. - <i>PWM A low & B high</i>: Set PWM A low and B high. - <i>No action</i>: No action taken.
Peak-to-Peak Value	Peak-to-peak value V_{pp} of the carrier wave
Offset Value	DC offset value V_{offset} of the carrier wave
Initial Input Value u, v, w	Initial value of 3-phase inputs u , v , and w
Start PWM at Beginning	When it is set to <i>Start</i> , PWM will start right from the beginning. If it is set to <i>Do not start</i> , one needs to start PWM using the Start PWM block.
Simulation Output Mode	The simulation output mode can be set to <i>Switching mode</i> or <i>Average mode</i> . When it is set to "Switching mode", the outputs of the PWM block are PWM signals. When it is set to "Average mode", the outputs of the PWM block are average mode signals. In the average mode, if the carrier wave is from negative to positive, and the absolute values of the negative peak and the positive peak are equal (for example, the carrier wave is from -1 to +1, or from -5 to +5), the modulation is considered as an ac signal modulation. Otherwise, the modulation is considered as a dc signal modulation. For example, modulation in a 3-phase or single-phase inverter is an ac modulation, and modulation in a buck converter is a dc modulation. In the ac signal modulation, if the input u of the PWM block is V_u , the output up and un in average mode will be: $V_{up} = V_u / (V_{pp} + V_{offset})$ $V_{un} = -V_{up}$ In this case, V_u is between $-(V_{pp} + V_{offset})$ and $V_{pp} + V_{offset}$, and V_{up} is between -1 to +1. In the dc signal modulation, the output up and un in average mode will be: $V_{up} = (V_u - V_{offset}) / V_{pp}$ $V_{un} = 1 - V_{up}$ In this case, V_u is between V_{offset} and $V_{pp} + V_{offset}$, and V_{up} is between 0 to +1. When it is set to the average mode, the PWM block outputs can be connected to a converter/inverter in the average mode model.

8.4.2 1-Phase PWM and 1-Phase PWM (phase shift)

The attributes for the **1-Phase PWM** block and **1-phase PWM (phase shift)** block are mostly the same. The difference is that the 1-Phase PWM block defines the phase shift through a parameter, while the 1-Phase PWM (phase shift) block reads the phase shift from an external input (labeled as "phase" in the image). The phase shift is in degree.

Image:



Attributes:

Parameters	Description
PWM Source	Source of the PWM generator. Without phase shift, it can be PWM 1 to PWM 8. With phase shift, it can be PWM 2 to PWM 8.
Output Mode	Output mode of the PWM generator. It can be one of the following: <ul style="list-style-type: none"> - <i>Use PWM A&B</i>: Both PWM outputs A and B are used, and they are complementary. - <i>Use PWM A</i>: Only PWM output A is used. - <i>Use PWM B</i>: Only PWM output B is used.
Dead Time	Dead time T_d for the PWM generator, in sec.
Sampling Frequency	Sampling frequency of the PWM generator, in Hz. The calculation is done and the PWM signal duty cycle is updated based on this frequency.
PWM Freq. Scaling Factor	The ratio between the PWM switching frequency and the sampling frequency. It can be from 1 to 100. For example, if the sampling frequency is 50 kHz and the scaling factor is 3, the PWM switching frequency will be 150 kHz. That is switches will operate at 150 kHz. But gating signals will be updated at 50 kHz, or once per 3 switching cycles.
Carrier Wave Type	Carrier wave type and the initial PWM output state. It can be one of the following: <ul style="list-style-type: none"> - <i>Triangular (start low)</i>: Triangular wave, and the initial PWM output state is low. - <i>Triangular (start high)</i>: Triangular wave, and the initial output state is high. - <i>Sawtooth (start low)</i>: Sawtooth wave, and the initial output state is low. - <i>Sawtooth (start high)</i>: Sawtooth wave, and the initial output state is high.
Trigger ADC	Setting whether for the PWM generator to trigger the A/D converter. It can be one of the following: <ul style="list-style-type: none"> - <i>Do not trigger ADC</i>: PWM does not trigger the A/D converter. - <i>Trigger ADC Group A</i>: PWM will trigger Group A of the A/D converter. - <i>Trigger ADC Group B</i>: PWM will trigger Group B of the A/D converter. - <i>Trigger ADC Group A&B</i>: PWM will trigger both Group A and B of the A/D converter.
ADC Trigger Position	A/D trigger position ranges from 0 to a value less than 1. When it is 0, the A/D converter is triggered at the beginning of the PWM cycle, and when it is 0.5, the A/D converter is triggered at the 180° position of the PWM cycle.
Use Trip-Zone i	Define whether the PWM generator uses the i_{th} trip-zone signal or not, where i ranges from 1 to 6. It can be one of the following: <ul style="list-style-type: none"> - <i>Disable Trip-Zone i</i>: Disable the i_{th} trip-zone signal. - <i>One shot</i>: The PWM generator uses the trip-zone signal in the one-shot mode. Once triggered, the PWM must be started manually. - <i>Cycle by cycle</i>: The PWM generator uses the trip-zone signal in the cycle-by-cycle basis. The trip-zone signal is effective within the current cycle, and PWM will automatically re-start in the next cycle.

PWMA DC Trip Src1(DCAH)	Digital compare (DC) trip source DCAH for PWMA. The PWM channel may have up to two DC trip sources: DCAH and DCAL. The trip source can be one of the following: <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>Trip-zone 1</i>: PWM uses trip-zone 1 as digital compare input signal. - <i>Trip-zone 2</i>: PWM uses trip-zone 2 as digital compare input signal. - <i>Trip-zone 3</i>: PWM uses trip-zone 3 as digital compare input signal. - <i>Comparator 1</i>: PWM uses Comparator 1 as digital compare input signal. - <i>Comparator 2</i>: PWM uses Comparator 2 as digital compare input signal. - <i>Comparator 3</i>: PWM uses Comparator 3 as digital compare input signal.
PWMA DC Trip Src1(DCAL)	Digital compare (DC) trip source DCAL for PWMA. The trip source can be one of the following: <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>Trip-zone 1</i>: PWM uses trip-zone 1 as digital compare input signal. - <i>Trip-zone 2</i>: PWM uses trip-zone 2 as digital compare input signal. - <i>Trip-zone 3</i>: PWM uses trip-zone 3 as digital compare input signal. - <i>Comparator 1</i>: PWM uses Comparator 1 as digital compare input signal. - <i>Comparator 2</i>: PWM uses Comparator 2 as digital compare input signal. - <i>Comparator 3</i>: PWM uses Comparator 3 as digital compare input signal.
PWMA 1-shot Evt (DCAEVT1)	Define how the one-shot signal is used for the DC trip signal of PWMA. The active level of the DC trip signal can be selected from the following: <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>Trip source 1 is low</i>: PWM is tripped if the source signal 1 is low. - <i>Trip source 1 is high</i>: PWM is tripped if the source signal 1 is high. - <i>Trip source 2 is low</i>: PWM is tripped if the source signal 2 is low. - <i>Trip source 3 is high</i>: PWM is tripped if the source signal 2 is high. - <i>Trip source 1 low & source 2 high</i>: PWM is tripped if the source 1 signal is low and at the same time source 2 signal is high.
PWMA CBC Evt (DCAEVT1)	Define how the cycle-by-cycle signal is used for the DC trip signal of PWMA. The active level of the DC trip signal can be selected from the following: <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>Trip source 1 is low</i>: PWM is tripped if the source signal 1 is low. - <i>Trip source 1 is high</i>: PWM is tripped if the source signal 1 is high. - <i>Trip source 2 is low</i>: PWM is tripped if the source signal 2 is low. - <i>Trip source 3 is high</i>: PWM is tripped if the source signal 2 is high. - <i>Trip source 1 low & source 2 high</i>: PWM is tripped if the source 1 signal is low and at the same time source 2 signal is high.
PWMB DC Trip Src1(DCBH)	Digital compare (DC) trip source DCBH for PWMB. The PWM channel may have up to two DC trip sources: DCBH and DCBL. The trip source can be one of the following: <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>Trip-zone 1</i>: PWM uses trip-zone 1 as digital compare input signal. - <i>Trip-zone 2</i>: PWM uses trip-zone 2 as digital compare input signal. - <i>Trip-zone 3</i>: PWM uses trip-zone 3 as digital compare input signal. - <i>Comparator 1</i>: PWM uses Comparator 1 as digital compare input signal. - <i>Comparator 2</i>: PWM uses Comparator 2 as digital compare input signal. - <i>Comparator 3</i>: PWM uses Comparator 3 as digital compare input signal.

PWMB DC Trip Src1(DCBL)	Digital compare (DC) trip source DCBL for PWMB. The trip source can be one of the following: <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>Trip-zone 1</i>: PWM uses trip-zone 1 as digital compare input signal. - <i>Trip-zone 2</i>: PWM uses trip-zone 2 as digital compare input signal. - <i>Trip-zone 3</i>: PWM uses trip-zone 3 as digital compare input signal. - <i>Comparator 1</i>: PWM uses Comparator 1 as digital compare input signal. - <i>Comparator 2</i>: PWM uses Comparator 2 as digital compare input signal. - <i>Comparator 3</i>: PWM uses Comparator 3 as digital compare input signal.
PWMB 1-shot Evt (DCBEVT1)	Define how the one-shot signal is used for the DC trip signal of PWMB. The active level of the DC trip signal can be selected from the following: <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>Trip source 1 is low</i>: PWM is tripped if the source signal 1 is low. - <i>Trip source 1 is high</i>: PWM is tripped if the source signal 1 is high. - <i>Trip source 2 is low</i>: PWM is tripped if the source signal 2 is low. - <i>Trip source 3 is high</i>: PWM is tripped if the source signal 2 is high. - <i>Trip source 1 low & source 2 high</i>: PWM is tripped if the source 1 signal is low and at the same time source 2 signal is high.
PWMB CBC Evt (DCBEVT1)	Define how the cycle-by-cycle signal is used for the DC trip signal of PWMB. The active level of the DC trip signal can be selected from the following: <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>Trip source 1 is low</i>: PWM is tripped if the source signal 1 is low. - <i>Trip source 1 is high</i>: PWM is tripped if the source signal 1 is high. - <i>Trip source 2 is low</i>: PWM is tripped if the source signal 2 is low. - <i>Trip source 3 is high</i>: PWM is tripped if the source signal 2 is high. - <i>Trip source 1 low & source 2 high</i>: PWM is tripped if the source 1 signal is low and at the same time source 2 signal is high.
DC Event Filter Source	Source of the digital compare event filter. It can be one of the following: <ul style="list-style-type: none"> - <i>Do not use</i>: Do not use DC event filter. - <i>DCAEVT1</i>: Use DCAEVT1 as the filter source. - <i>DCAEVT2</i>: Use DCAEVT2 as the filter source. - <i>DCBEVT1</i>: Use DCBEVT1 as the filter source. - <i>DCBEVT2</i>: Use DCBEVT2 as the filter source. - <i>DCAEVT2</i>: PWM is tripped if the source signal 1 is high.
Blanking Window Pos (us)	Blanking window start position in a PWM period, in us.
Blanking Window Width (us)	Width of the blanking window, in us. The width is limited by the hardware, and can be calculated as below: $255 / \text{CPU Frequency}$ <p>For example, when the CPU speed is 90MHz, the width range is 0 to 2.83us.</p>
Blanking Window Range	Specify how the blanking window is applied. It can be one of the following: <ul style="list-style-type: none"> - <i>In the window</i>: The blanking action is applied with the window defined (from the start position for a window width defined) - <i>Out of window</i>: The blanking action is applied outside the window defined.
Applying Event Filtering	Specify how event filtering is applied to digital compare events. The event filtering can be applied to any combinations of the following digital compare events: DCAEVT1, DCAEVT2, DCBEVT1, and DCBEVT2

Trip Action	Define how the PWM generator responds to the trip action. It can be one of the following: <ul style="list-style-type: none"> - <i>High impedance</i>: PWM outputs in high impedance - <i>PWM A high & B low</i>: Set PWM A high and B low. - <i>PWM A low & B high</i>: Set PWM A low and B high. - <i>No action</i>: No action taken.
Peak-to-Peak Value	Peak-to-peak value V_{pp} of the carrier wave
Offset Value	DC offset value V_{offset} of the carrier wave
Phase Shift	Phase shift of the output with respect to the reference PWM generator output, in deg. (for 1-phase PWM generator without external phase shift input)
Initial Input Value	Initial value of the input
Use HRPWM	Define the high-resolution PWM. It can be one of the following: <ul style="list-style-type: none"> - <i>Do not use HRPWM</i>: Do not use high-resolution PWM - <i>Use HRPWM without calibration</i>: Use high-resolution PWM without calibration - <i>Use HRPWM with calibration</i>: Use high-resolution PWM with calibration
Start PWM at Beginning	When it is set to <i>Start</i> , PWM will start right from the beginning. If it is set to <i>Do not start</i> , one needs to start PWM using the Start PWM block.
Simulation Output Mode	The simulation output mode can be set to <i>Switching mode</i> or <i>Average mode</i> . When it is set to "Switching mode", the outputs of the PWM block are PWM signals. When it is set to "Average mode", the outputs of the PWM block are average mode signals. In the average mode, if the carrier wave is from negative to positive, and the absolute values of the negative peak and the positive peak are equal (for example, the carrier wave is from -1 to +1, or from -5 to +5), the modulation is considered as an ac signal modulation. Otherwise, the modulation is considered as a dc signal modulation. For example, modulation in a 3-phase or single-phase inverter is an ac modulation, and modulation in a buck converter is a dc modulation. In the ac signal modulation, if the input of the PWM block is V_{in} , the output A and B in average mode will be: $V_A = V_{in} / (V_{pp} + V_{offset})$ $V_B = -V_A$ In this case, V_{in} is between $-(V_{pp} + V_{offset})$ and $V_{pp} + V_{offset}$, and V_A and V_B are between -1 to +1. In the dc signal modulation, the output A and B in average mode will be: $V_A = (V_{in} - V_{offset}) / V_{pp}$ $V_B = 1 - V_A$ In this case, V_{in} is between V_{offset} and $V_{pp} + V_{offset}$, and V_A and V_B are between 0 to +1. When it is set to the average mode, the PWM block outputs can be connected to a converter/inverter in the average mode model.

Phase Shift

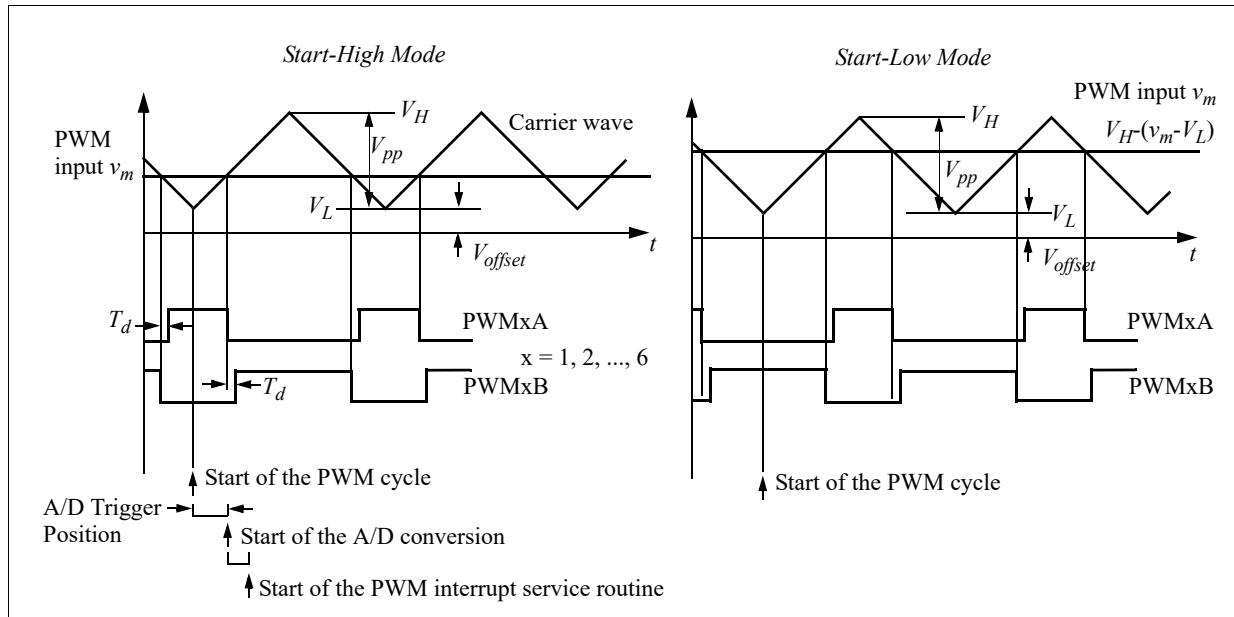
A 1-phase PWM generator can generate PWM signal that is phase shifted with respect to another PWM signal. The way how PWM blocks are defined for phase shift is explained in Section 8.4.5.

The phase shift value is in degrees. When the value is -30° , the output will be shifted to the right (lagging) by 30° of the switching cycle with respect to the reference PWM generator output. This is equivalent to shifting the

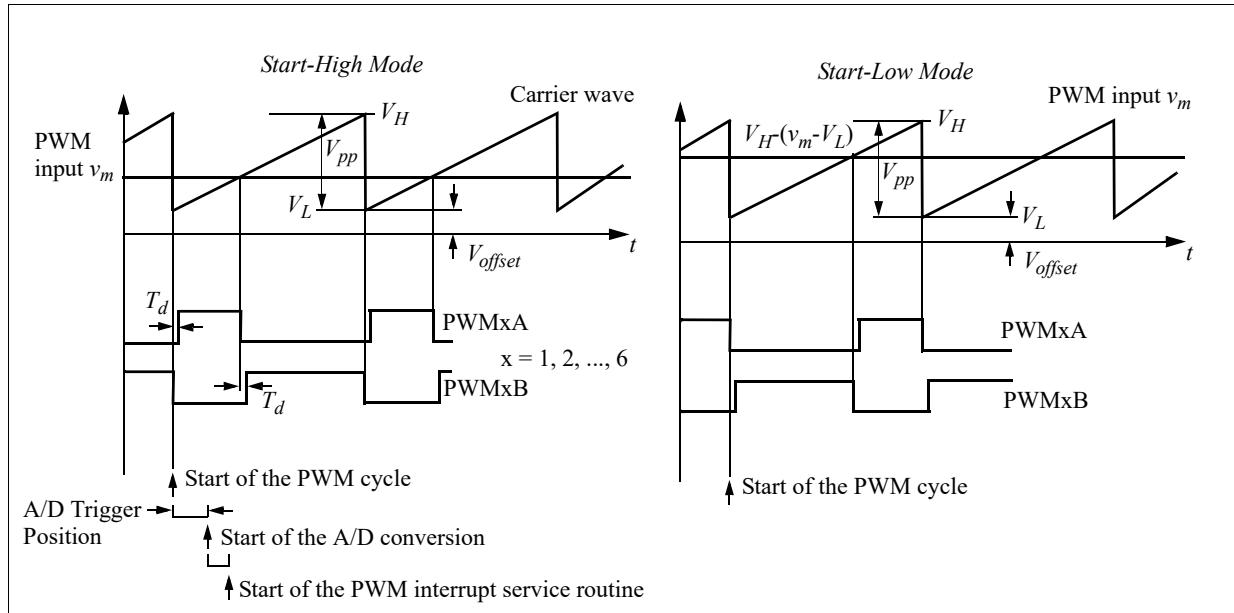
PWM carrier wave to the right by 30° . When the phase value is 30° , the output will be shifted to the left (leading) by 30° .

Carrier Wave

There are two types of carrier waveforms: triangular wave (with equal rising and falling slope intervals) and sawtooth wave. In addition, there are two operation modes: start-low and start-high modes, as explained below. The input and output waveforms of a PWM generator with the triangular carrier wave are shown below:



The input and output waveforms of a PWM generator with the sawtooth carrier wave are shown below:



The figures above show how the dead time is defined, and the time sequence when the PWM generator triggers the A/D converter. If triggering the A/D converter is selected, from the start of the PWM cycle, after a certain delay defined by the A/D trigger position, the A/D conversion will start. After the A/D conversion is completed, the PWM interrupt service routine will start.

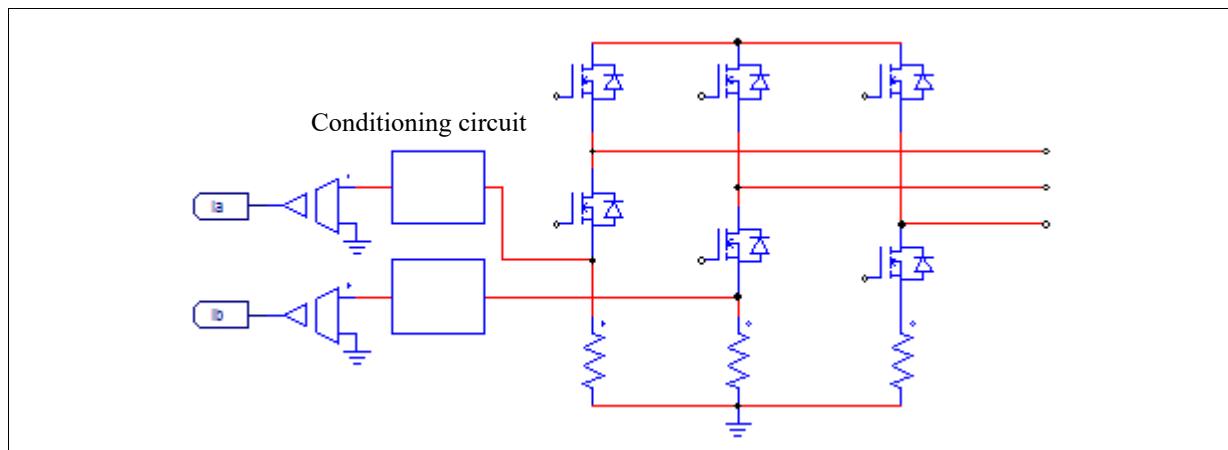
If the PWM generator does not trigger the A/D converter, the PWM interrupt service routine will start at the beginning of the PWM cycle.

The figures above also show how the start-high and start-low modes work. Assume that the PWM input is v_m , and the lowest value of the carrier wave is V_L and the highest value is V_H . In the start-high mode, the PWM positive output PWMA is high at the beginning of the switching cycle, and it remains high as long as the input v_m is greater than the carrier wave. For example, for a carrier wave from 0 to 1, $V_L=0$, and $V_H=1$. If $v_m=0.2$, the PWM output PWMA will remain high as long as the carrier is less than 0.2.

On the other hand, in the start-low mode, the PWM positive output PWMA is low at the beginning of the switching cycle, and it is high when the carrier wave is greater than the value $V_H(v_m-V_L)$. For example, for a carrier wave from 0 to 1, $V_L=0$, and $V_H=1$. If $v_m=0.2$, the PWM output PWMA will be high as long as the carrier is greater than 0.8.

The carrier start mode depends on how switch currents are measured. In a 3-phase inverter, for example, if top switch currents are measured, the start-high mode should be selected. This ensures that at the beginning of the cycle, the top switch gating signal is high and the current is conducting. On the other hand, if bottom switch currents are measured, the start-low mode should be selected. This ensures that at the beginning of the cycle, the top switch gating signal is low, and the bottom switch gating signal is high and the current is conducting.

For example, in the circuit below, the bottom switch currents of Phase A and B are measured. In this case, the carrier start-low mode should be selected.

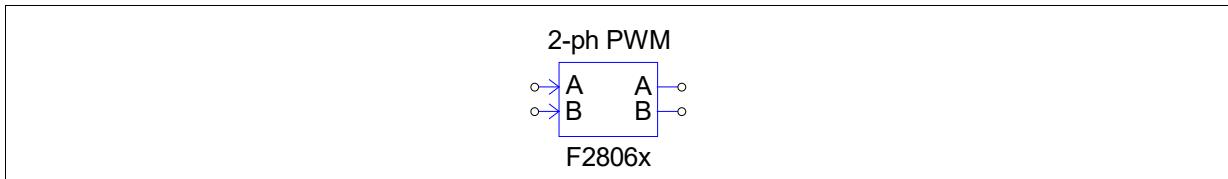


Note: In the start-low mode, the PWM input v_m is converted to $V_H(v_m-V_L)$ internally before it is compared with the carrier wave to generate the PWM signal. With the conversion, both the start-low and start-high modes will have the same duty cycle expression. For example, for a sawtooth wave with $V_L=0$ and $V_H=1$, or for a triangular wave with $V_L= -V_H$, the duty cycle D of the PWMA output in both the start-low and start-high modes is: $D = v_m/V_H$.

8.4.3 2-Phase PWM Generator

A 2-phase PWM generator has two inputs and two outputs, and operates in one of the six pre-defined operation modes.

Image:



Attributes:

Parameters	Description
PWM Source	Source of the PWM generator. It can be PWM 1 to PWM 8.
Mode Type	Operation mode of the PWM generation. It can be one of the 6 modes. The waveforms of the 6 operation modes are described below.
Sampling Frequency	Sampling frequency of the PWM generator, in Hz. The calculation is done and the PWM signal duty cycle is updated based on this frequency.
PWM Freq. Scaling Factor	The ratio between the PWM switching frequency and the sampling frequency. It can be from 1 to 100. For example, if the sampling frequency is 50 kHz and the scaling factor is 3, the PWM switching frequency will be 150 kHz. That is switches will operate at 150 kHz. But gating signals will be updated at 50 kHz, or once per 3 switching cycles.
Trigger ADC	Setting whether for the PWM generator to trigger the A/D converter. It can be one of the following: <ul style="list-style-type: none"> - <i>Do not trigger ADC</i>: PWM does not trigger the A/D converter. - <i>Trigger ADC Group A</i>: PWM will trigger Group A of the A/D converter. - <i>Trigger ADC Group B</i>: PWM will trigger Group B of the A/D converter. - <i>Trigger ADC Group A&B</i>: PWM will trigger both Group A and B of the A/D converter.
ADC Trigger Position	A/D trigger position ranges from 0 to a value less than 1. When it is 0, the A/D converter is triggered at the beginning of the PWM cycle, and when it is 0.5, the A/D converter is triggered at the 180° position of the PWM cycle.
Use Trip-Zone <i>i</i>	Define whether the PWM generator uses the <i>i</i> th trip-zone signal or not, where <i>i</i> ranges from 1 to 6. It can be one of the following: <ul style="list-style-type: none"> - <i>Disable Trip-Zone <i>i</i></i>: Disable the <i>i</i>th trip-zone signal. - <i>One shot</i>: The PWM generator uses the trip-zone signal in the one-shot mode. Once triggered, the PWM must be started manually. - <i>Cycle by cycle</i>: The PWM generator uses the trip-zone signal in the cycle-by-cycle basis. The trip-zone signal is effective within the current cycle, and PWM will automatically re-start in the next cycle.
PWMA DC Trip Src1(DCAH)	Digital compare (DC) trip source DCAH for PWMA. The PWM channel may have up to two DC trip sources: DCAH and DCAL. The trip source can be one of the following: <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>Trip-zone 1</i>: PWM uses trip-zone 1 as digital compare input signal. - <i>Trip-zone 2</i>: PWM uses trip-zone 2 as digital compare input signal. - <i>Trip-zone 3</i>: PWM uses trip-zone 3 as digital compare input signal. - <i>Comparator 1</i>: PWM uses Comparator 1 as digital compare input signal. - <i>Comparator 2</i>: PWM uses Comparator 2 as digital compare input signal. - <i>Comparator 3</i>: PWM uses Comparator 3 as digital compare input signal.
PWMA DC Trip Src1(DCAL)	Digital compare (DC) trip source DCAL for PWMA. The trip source can be one of the following: <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>Trip-zone 1</i>: PWM uses trip-zone 1 as digital compare input signal. - <i>Trip-zone 2</i>: PWM uses trip-zone 2 as digital compare input signal. - <i>Trip-zone 3</i>: PWM uses trip-zone 3 as digital compare input signal. - <i>Comparator 1</i>: PWM uses Comparator 1 as digital compare input signal. - <i>Comparator 2</i>: PWM uses Comparator 2 as digital compare input signal. - <i>Comparator 3</i>: PWM uses Comparator 3 as digital compare input signal.

PWMA 1-shot Evt (DCAEVT1)	Define how the one-shot signal is used for the DC trip signal of PWMA. The active level of the DC trip signal can be selected from the following: <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>Trip source 1 is low</i>: PWM is tripped if the source signal 1 is low. - <i>Trip source 1 is high</i>: PWM is tripped if the source signal 1 is high. - <i>Trip source 2 is low</i>: PWM is tripped if the source signal 2 is low. - <i>Trip source 3 is high</i>: PWM is tripped if the source signal 2 is high. - <i>Trip source 1 low & source 2 high</i>: PWM is tripped if the source 1 signal is low and at the same time source 2 signal is high.
PWMA CBC Evt (DCAEVT1)	Define how the cycle-by-cycle signal is used for the DC trip signal of PWMA. The active level of the DC trip signal can be selected from the following: <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>Trip source 1 is low</i>: PWM is tripped if the source signal 1 is low. - <i>Trip source 1 is high</i>: PWM is tripped if the source signal 1 is high. - <i>Trip source 2 is low</i>: PWM is tripped if the source signal 2 is low. - <i>Trip source 3 is high</i>: PWM is tripped if the source signal 2 is high. - <i>Trip source 1 low & source 2 high</i>: PWM is tripped if the source 1 signal is low and at the same time source 2 signal is high.
PWMB DC Trip Src1(DCBH)	Digital compare (DC) trip source DCBH for PWMB. The PWM channel may have up to two DC trip sources: DCBH and DCBL. The trip source can be one of the following: <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>Trip-zone 1</i>: PWM uses trip-zone 1 as digital compare input signal. - <i>Trip-zone 2</i>: PWM uses trip-zone 2 as digital compare input signal. - <i>Trip-zone 3</i>: PWM uses trip-zone 3 as digital compare input signal. - <i>Comparator 1</i>: PWM uses Comparator 1 as digital compare input signal. - <i>Comparator 2</i>: PWM uses Comparator 2 as digital compare input signal. - <i>Comparator 3</i>: PWM uses Comparator 3 as digital compare input signal.
PWMB DC Trip Src1(DCBL)	Digital compare (DC) trip source DCBL for PWMB. The trip source can be one of the following: <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>Trip-zone 1</i>: PWM uses trip-zone 1 as digital compare input signal. - <i>Trip-zone 2</i>: PWM uses trip-zone 2 as digital compare input signal. - <i>Trip-zone 3</i>: PWM uses trip-zone 3 as digital compare input signal. - <i>Comparator 1</i>: PWM uses Comparator 1 as digital compare input signal. - <i>Comparator 2</i>: PWM uses Comparator 2 as digital compare input signal. - <i>Comparator 3</i>: PWM uses Comparator 3 as digital compare input signal.
PWMB 1-shot Evt (DCBEVT1)	Define how the one-shot signal is used for the DC trip signal of PWMB. The active level of the DC trip signal can be selected from the following: <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>Trip source 1 is low</i>: PWM is tripped if the source signal 1 is low. - <i>Trip source 1 is high</i>: PWM is tripped if the source signal 1 is high. - <i>Trip source 2 is low</i>: PWM is tripped if the source signal 2 is low. - <i>Trip source 3 is high</i>: PWM is tripped if the source signal 2 is high. - <i>Trip source 1 low & source 2 high</i>: PWM is tripped if the source 1 signal is low and at the same time source 2 signal is high.

PWMB CBC Evt (DCBEVT1)	Define how the cycle-by-cycle signal is used for the DC trip signal of PWMB. The active level of the DC trip signal can be selected from the following: <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>Trip source 1 is low</i>: PWM is tripped if the source signal 1 is low. - <i>Trip source 1 is high</i>: PWM is tripped if the source signal 1 is high. - <i>Trip source 2 is low</i>: PWM is tripped if the source signal 2 is low. - <i>Trip source 3 is high</i>: PWM is tripped if the source signal 2 is high. - <i>Trip source 1 low & source 2 high</i>: PWM is tripped if the source 1 signal is low and at the same time source 2 signal is high.
DC Event Filter Source	Source of the digital compare event filter. It can be one of the following: <ul style="list-style-type: none"> - <i>Do not use</i>: Do not use DC event filter. - <i>DCAEVT1</i>: Use DCAEVT1 as the filter source. - <i>DCAEVT2</i>: Use DCAEVT2 as the filter source. - <i>DCBEVT1</i>: Use DCBEVT1 as the filter source. - <i>DCBEVT2</i>: Use DCBEVT2 as the filter source. - <i>DCAEVT2</i>: PWM is tripped if the source signal 1 is high.
Blanking Window Pos (us)	Blanking window start position in a PWM period, in us.
Blanking Window Width (us)	Width of the blanking window, in us. The width is limited by the hardware, and can be calculated as below: $255 / \text{CPU Frequency}$ <p>For example, when the CPU speed is 90MHz, the width range is 0 to 2.83us.</p>
Blanking Window Range	Specify how the blanking window is applied. It can be one of the following: <ul style="list-style-type: none"> - <i>In the window</i>: The blanking action is applied with the window defined (from the start position for a window width defined) - <i>Out of window</i>: The blanking action is applied outside the window defined.
Applying Event Filtering	Specify how event filtering is applied to digital compare events. The event filtering can be applied to any combinations of the following digital compare events: DCAEVT1, DCAEVT2, DCBEVT1, and DCBEVT2
Trip Action	Define how the PWM generator responds to the trip action. It can be one of the following: <ul style="list-style-type: none"> - <i>High impedance</i>: PWM outputs in high impedance - <i>PWM A high & B low</i>: Set PWM A high and B low. - <i>PWM A low & B high</i>: Set PWM A low and B high. - <i>No action</i>: No action taken.
Peak Value	Peak value V_{pk} of the carrier wave
Initial Input Value A, B	Initial value of the inputs A and B.
Start PWM at Beginning	When it is set to "Start", PWM will start right from the beginning. If it is set to "Do not start", one needs to start PWM using the "Start PWM" function.

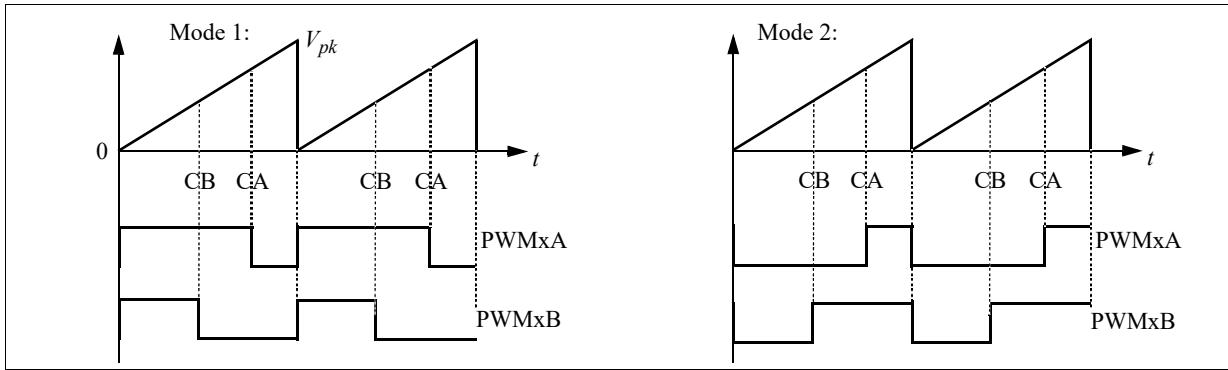
For 2-phase PWM generators, the outputs are determined based on the mode of operation, as described below. The carrier wave is either sawtooth or triangular, depending on the mode of operation. It increases from 0 to the peak value V_{pk} , and there is no dc offset.

Operation Mode 1:

The figure below on the left shows the waveforms of Mode 1. In the figure, "CA" and "CB" refer to two inputs A and B of the 2-phase PWM generator. Each input controls the turn-off time of each output.

Operation Mode 2:

The figure below on the right shows the waveforms of Mode 2. Unlike in Mode 1, each input controls the turn-on time of each output.

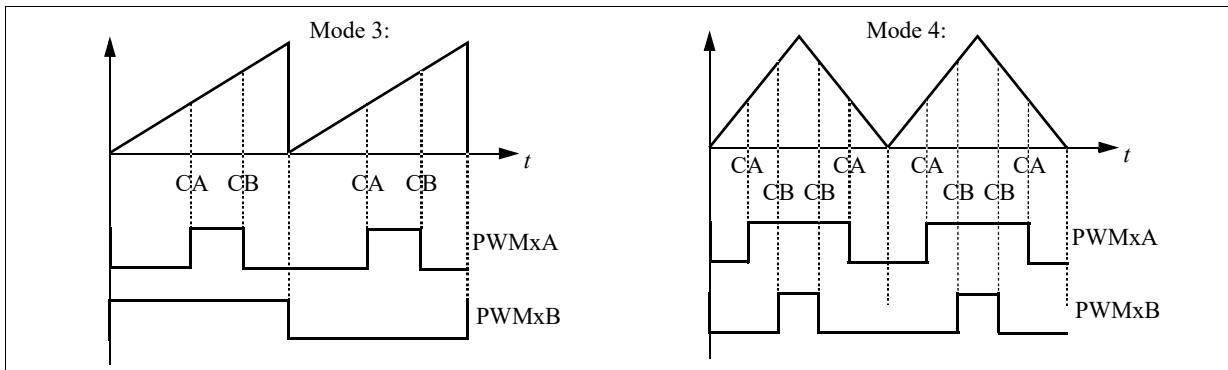


Operation Mode 3:

The figure below on the left shows the waveforms of Mode 3. Input A controls the turn-on and Input B controls the turn-off of the PWM output A. The PWM output B is on for one complete PWM cycle, and is off for the next cycle.

Operation Mode 4:

The figure below on the right shows the waveforms of Mode 4. The carrier wave is triangular. Each input controls both the turn-on and turn-off of its output.

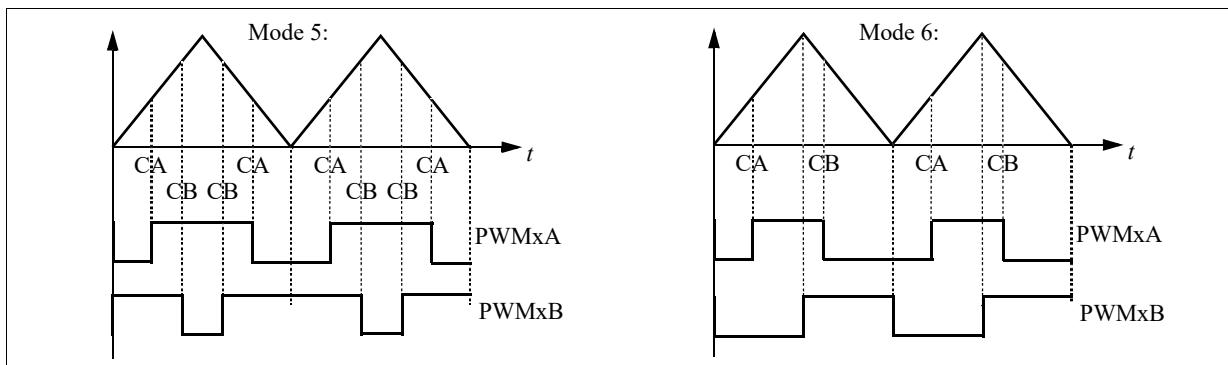


Operation Mode 5:

The figure below on the left shows the waveforms of Mode 5. The carrier wave is triangular. Similar to Mode 4, each input controls both the turn-on and turn-off of its output. Note that PWM output B is inverted in this case.

Operation Mode 6:

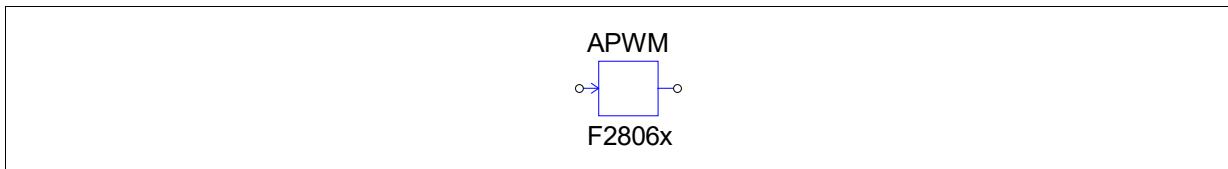
The figure below on the right shows the waveforms of Mode 6. In this mode, Input A controls the turn-on and Input B controls the turn-off of PWM output A. The PWM output B is on for the first half PWM cycle, and is off for the second half cycle.



8.4.4 Single PWM (shared with capture)

A single PWM generator, also called APWM, shares the same resource as captures. It has restricted functionality as compared to the 6 PWM generators (PWM 1 to 6) as they cannot trigger the A/D converter and cannot use trip-zone signals. Also, because of the common resource, when a particular port is used for the capture, it can not be used for the PWM generator.

Image:



Attributes:

Parameters	Description
PWM Source	Source of the PWM generator. It can be from one of the following: <ul style="list-style-type: none">- APWM 1 (GPIO 5, 11, 19, and 24)- APWM 2 (GPIO 7, 15, and 25)- APWM 3 (GPIO 9 and 26)
PWM Frequency	Frequency of the PWM generator, in Hz
Carrier Wave Type	The carrier wave type and the initial PWM output state. It can be one of the following: <ul style="list-style-type: none">- <i>Sawtooth (start low)</i>: Sawtooth wave, with the PWM output in the low state initially.- <i>Sawtooth (start high)</i>: Sawtooth wave, with the PWM output in the high state initially.
Stop Action	The output status when the PWM generator is stopped. It can be one of the following: <ul style="list-style-type: none">- <i>Output low</i>: The PWM output will be set to low.- <i>Output high</i>: The PWM output will be set to high.
Peak-to-Peak Value	Peak-to-peak value of the carrier wave
Offset Value	DC offset value of the carrier wave
Phase Shift	Phase shift of the output with respect to the reference PWM generator, in deg.
Initial Input Value	Initial value of the input
Start PWM at Beginning	When it is set to <i>Start</i> , PWM will start right from the beginning. If it is set to <i>Do not start</i> , one needs to start PWM using the Start PWM block.

Similar to 1-phase PWM generators, an APWM generator can generate a PWM signal that has a phase shift with respect to another PWM generator. The way how PWM blocks are defined for phase shift is explained in Section 8.4.5.

8.4.5 Synchronization Between PWM Blocks

Three types of PWM blocks can be synchronized, and phase shifts can be defined between each other: **1-phase PWM**, **1-phase PWM (phase shift)**, and **APWM (or Single PWM (shared with capture))**.

A 1-phase PWM block can generate PWM signal that is phase shifted with respect to another PWM signal. There is one series in regular PWM blocks: PWM 1, 2, 3, 4, 5, 6, 7, and 8.

Similarly, there is one series in APWM blocks for phase shift: PWM 1, APWM 1, 2, and 3.

The definitions of the PWM blocks for phase shift are described below.

- The reference PWM and the PWM being phase shifted must be from the same series. That is, PWM 1 can

be the reference, and PWM 2, 3, 4, 5, 6, 7, and 8 can be phase shifted with respect to PWM 1. Or PWM 2 can be the reference, and PWM 3 to 8 can be phase shifted with respect to PWM 2.

This also applies to APWM blocks. For example, PWM 1 can be the reference, and APWM 1, 2, and 3 can be phase shifted. Or APWM 1 can be the reference, and APWM 2 and 3 can be phase shifted.

- The reference PWM and the PWM being shifted must be consecutive in the series, unless the skipped PWM is not used. For example, if PWM 1, 2, and 3 are all used, but PWM 2 is not synchronized with PWM 1 and 3, this is not allowed. However, if PWM 2 is not used in the circuit, it is ok to use PWM 1 as the reference and phase shift PWM 3.

For example, the following definitions are correct, with the first PWM as the reference and the subsequent PWM blocks phase shifted:

PWM 1 (reference), PWM 2, PWM 3, PWM 4, PWM 5, PWM 6, PWM 7, PWM 8

PWM 2 (reference), PWM 3

PWM 4 (reference), PWM 5

PWM 5 (reference), PWM 6

PWM 6 (reference), PWM 7

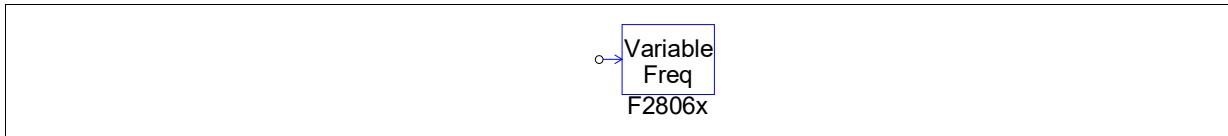
PWM 1 (reference), PWM 2, PWM 3, PWM 4, PWM 5, PWM 6, PWM 7, PWM 8, APWM 1, APWM 2, APWM 3

PWM 1 (reference), APWM 1, APWM 2, APWM 3

8.5 Variable Frequency PWM

The Variable Frequency PWM block provides the function to change the sampling frequency of a PWM generator. The image and parameters are shown below.

Image:



Attributes:

Parameters	Description
PWM Source	Source of the PWM generator. It can be one of the following: PWM 1, PWM 2, PWM 3, PWM 4, PWM 5, PWM 6, 3-phase PWM 123, and 3-phase PWM 456.
Adjust Interrupt Pos.	Specify if the interrupt position is adjusted with the frequency. It can be one of the following: - <i>Do not adjust</i> : The interrupt position will remain unchanged as calculated with the base frequency. - <i>Adjust</i> : The interrupt position will be recalculated at the beginning of each cycle based on the new frequency.
Adjust Ramp Compensation	Specify if the ramp compensation of the comparator DAC block is adjusted with the frequency. It can be one of the following: - <i>Do not adjust</i> : The ramp compensation will remain unchanged as calculated with the base frequency. - <i>Adjust</i> : The ramp compensation will be recalculated at the beginning of each cycle based on the new frequency.

The sampling frequency of the corresponding PWM block will be changed at the beginning of the next PWM period as follows:

$$\text{PWM_Frequency} = \text{PWM_Base Frequency} / \text{Input_Value}$$

where *PWM_Base_Frequency* is the sampling frequency of the corresponding PWM block, and *Input_Value* is the input value of this block.

If the interrupt position is to be adjusted, the interrupt position will be recalculated in each cycle. Since adjusting the interrupt position takes time, if the frequency change is small, it is recommended not to adjust the interrupt position.

Similarly, if the ramp compensation is to be adjusted, the ramp compensation will be recalculated in each cycle. Since adjusting the ramp compensation takes time, if the frequency change is small, it is recommended not to adjust the ramp compensation.

8.6 Start PWM and Stop PWM

The Start PWM and Stop PWM blocks provide the function to start/stop a PWM generator. The images and parameters are shown below.

Image:



Attributes:

Parameters	Description
PWM Source	The source of the PWM generator. It can be: PWM 1-7, 3-phase PWM 123 and PWM 456, and Capture 1.

8.7 Trip-Zone and Trip-Zone State

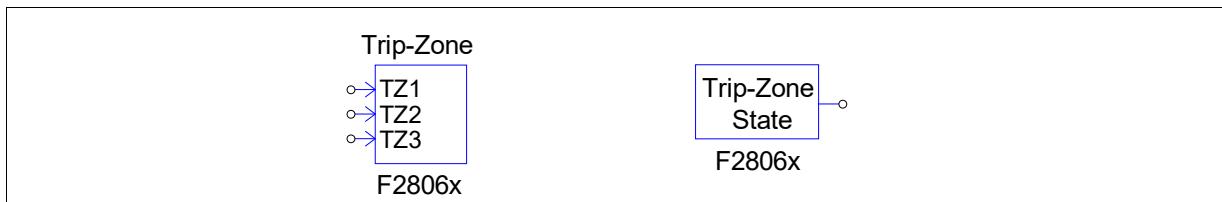
F2806x contains 6 trip-zone input signals: 3 from GPIO ports and 3 from comparators. Comparators can only be used in Digital Compare to trip PWM.

Trip-zone is used to handle external fault or trip conditions. The corresponding PWM outputs can be programmed to respond accordingly.

One trip-zone signal can be used by multiple PWM generators, and a PWM generator can use any or all of the 6 trip-zone signals. The interrupt generated by trip-zone signals are handled by the interrupt block.

The trip-zone signal triggers a trip action when the input signal is low (0). The trip-zone signals through Digital Compare trigger a trip action in specified level (either high or low)

Image:



Attributes for Trip-Zone:

Parameters	Description
Use Trip-Zone i	Specify if the i_{th} trip-zone is used.
GPIO Port for Trip-Zone i	Specify a designated GPIO port as the i_{th} trip-zone input signal. It can be one of the following: - For trip-zone 1: GPIO12 or 13 - For trip-zone 2: GPIO13, 16, or 18 - For trip-zone 3: GPIO14, 17, or 19
Use Comparator i	Specify if the i_{th} comparator is used as the trip-zone input signal

Attributes for Trip-Zone State:

Parameters	Description
PWM Source	Source of the PWM generator. It can be: PWM 1-7, 3-phase PWM 123, and PWM 456.

The trip-zone interrupt can be generated in either one-shot mode or cycle-by-cycle mode, as defined in the PWM generator parameter input. In the cycle-by-cycle mode, the interrupt only affects the PWM output within the current PWM cycle. On the other hand, in the one-shot mode, interrupt triggers a trip action when the input signal is low (0). will set the PWM output permanently, and the PWM generator must be restarted to resume the operation.

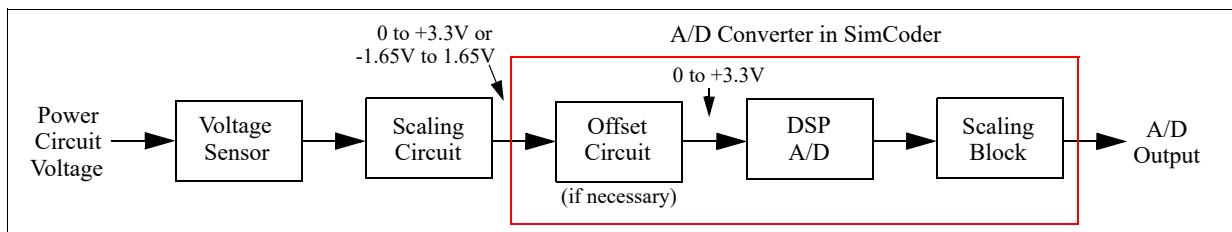
The Trip-Zone State element indicates whether the trip-zone signal is in one-shot mode or cycle-by-cycle mode when it triggers a PWM generator to generate an interrupt. When the output is 1, it means that the trip-zone signal is in one-shot mode. When the output is 0, the trip-zone signal is in cycle-by-cycle mode.

Note that when defining the interrupt block associate with trip-zone, the "Device Name" parameter of the interrupt block should be the name of the PWM generator, not the trip-zone block name. For example, if a PWM generator called "PWM_G1" uses trip-zone 1 in the trip-zone block "TZ1". The "Device Name" of the corresponding interrupt block should be "PWM_G1", not "TZ1". The "Channel Number" parameter in the interrupt block is not used in this case.

8.8 A/D Converter

F2806x provides a 16-channel 12-bit A/D converter.

Normally a power circuit quantity (voltage, current, speed, etc.) is brought to the DSP in several stages. For example, a power circuit voltage, which could be at a high level, is first converted to a control signal using a voltage sensor. A scaling circuit is then used to scale the signal, and an offset circuit is used to provide dc offset to the signal if necessary, so that the signal at the DSP A/D input is within the range of 0V and +3.3V. This signal is converted to a digital value in DSP, and a scaling block may be used to scale the value back to its original value. The complete process is shown in the diagram below.



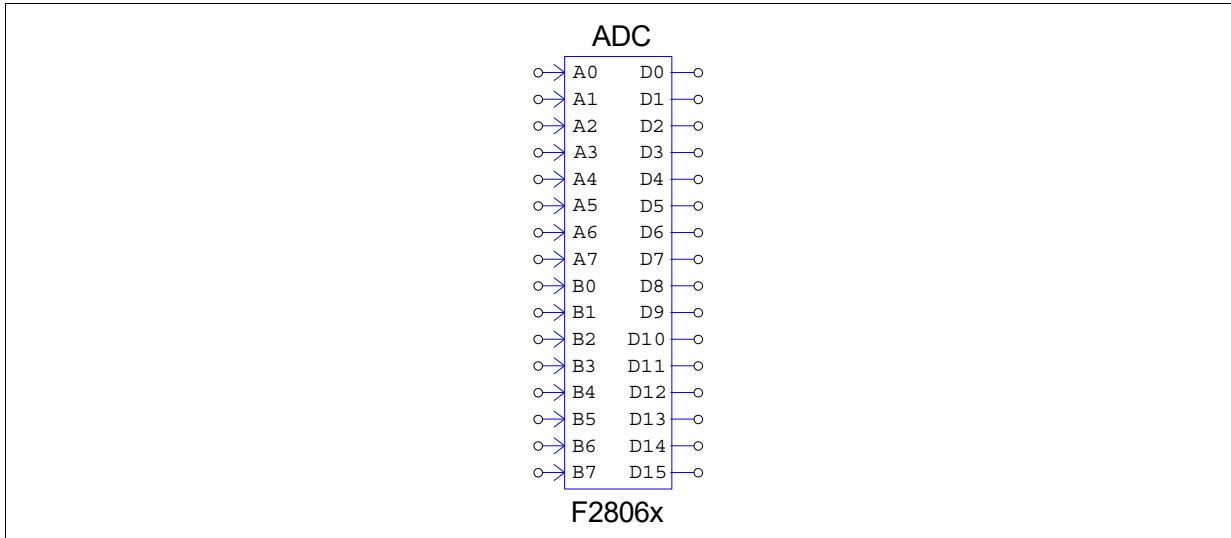
As shown above, the A/D converter element in SimCoder is not exactly the same as the physical A/D converter on the DSP. Rather, it combines the functions of an offset circuit, the DSP A/D converter, and a scaling block. This is designed for the convenience of AC system applications. It will be further explained in Section 6.5.3.

In many applications, the circuit variables to be monitored are AC signals, especially in AC motor drive systems. For each of this kind of AC signals, an offset circuit must be built in the hardware on circuit board at

the input of the DSP analog input, in order to shift the signal level to the acceptable range of 0 to +3.3V. SimCoder's A/D converter provides the convenience for such cases. Instead of level-shifting and scaling the A/D output signals, user may chose to use the offset option and scaling factor in the SimCoder A/D converter, and the target code will be generated accordingly.

The image and the parameters of the A/D converter in the SimCoder library are described below. In the following description, "A/D converter" refers to the A/D converter in the SimCoder library, not the DSP A/D converter, unless otherwise stated.

Image:



Attributes:

Parameters	Description
Ch A_i or B_i Mode	Input mode of the i_{th} A/D converter channel A_i or B_i . The input mode can be one of the following: - <i>AC</i> : This option is for simulation only, not for code generation. The input range is considered from -1.65V to +1.65V. This option includes the offset circuit into the A/D converter. It provides the convenience in cases where an external level shifter is needed to shift the AC signal to the 0 to +3.3V range. - <i>DC</i> : The input is a dc value, and the range is from 0 to +3.3V.
Ch A_i or B_i Gain	Gain k of the i_{th} A/D converter channel A_i or B_i .
Conversion Order	Order of the A/D conversion. If the field is left blank (undefined), the conversion will be done based on the serial numbers of the A/D channel. For example, if A0, A2, A4, B1, and B3 are used, the conversion will be done in this order: A0, A2, A4, B1, and B3. If you wish certain channels to be performed first, you can define the order here. For example, if the conversion order is defined as: A4,A0,A2,B3,B2 The conversion will be done in the order defined, that is, A4 before A0, and A0 before A2, and so on.
ADCINT1 PIE Selection	Specify if interrupt ADCINT1 uses <i>PIE Group1</i> or <i>PIE Group10</i> .
ADCINT2 PIE Selection	Specify if interrupt ADCINT2 uses <i>PIE Group1</i> or <i>PIE Group10</i> .

An A/D converter has up to 16 channels. SimCoder divides them into groups according their sampling rates. The group with the highest sampling rate uses interrupt ADCINT1, and the group with the second highest

sampling rate uses interrupt ADCINT2, etc. The two ADC groups with the highest sampling rates can choose interrupt from PIE (peripheral interrupt expansion) groups of the PIE vector table for different interrupt priority. PIE Group1 has a higher interrupt priority than PIE Group10. For example, PWM's interrupt is in PIE Group3, Its interrupt priority is lower than PIE Group1 but higher than Group10. If one wants PWM interrupt to have a higher priority than ADC interrupt, one needs to set the interrupt corresponding to the ADC channels to use PIE Group10.

Trigger Source:

The A/D converter can be triggered from multiple sources. Multiple A/D channels may share the same trigger source. Each A/D channel can be triggered by:

- One of the PWM generators,
- Timer1 or Timer2.
- More than one trigger source.

In a schematic, if a A/D channel is not associated with a PWM generator, one should insert a ZOH block at the output of the A/D channel so that SimCoder will select the timer as the trigger source.

It is not permitted to have a A/D converter channel triggered by one source, but its output signal is used in a circuit section that has a different sampling rate.

Output Scaling:

The output is scaled based on the following:

$$V_o = k * V_i$$

where V_i is the value at the input of the A/D converter.

Input Offset and Scaling:

The input of the A/D converter must stay within the input range. When the input is out of the range, it will be clamped to the limit, and a warning message will be given.

Also, the signal at the input port of the A/D converter must be scaled such that, when the input mode is DC, the maximum input voltage be scaled to +3.3V; and when the input mode is AC, the peak voltage be scaled to +/- 1.65V.

To illustrate how to use the A/D converter, two examples are given below: One with a dc input and the other with an ac input.

Assume that a power circuit voltage is a dc quantity, and the range is as follows:

$$V_{i_min} = 0 \text{ V}$$

$$V_{i_max} = 150 \text{ V}$$

The input mode of the A/D converter will be set to dc, and the input range is from 0 to +3.3V. Assume that the actual value of the voltage at a certain point is:

$$V_i = 100 \text{ V}$$

Let the voltage sensor gain be 0.01. After the voltage sensor, the maximum value and the actual value of the input become:

$$V_{i_max_s} = 150 * 0.01 = 1.5 \text{ V}$$

$$V_{i_s} = 100 * 0.01 = 1 \text{ V}$$

To utilize the full range of the DSP, a conditioning circuit with a gain of 2.2 will be used. The combined gain of the voltage sensor and the conditioning circuit becomes: $0.01 * 2.2 = 0.022$. After the conditioning circuit and at the input of the DSP A/D converter, the maximum value and the actual value of the input become:

$$V_{i_max_s_c} = 1.5 * 2.2 = 3.3 \text{ V}$$

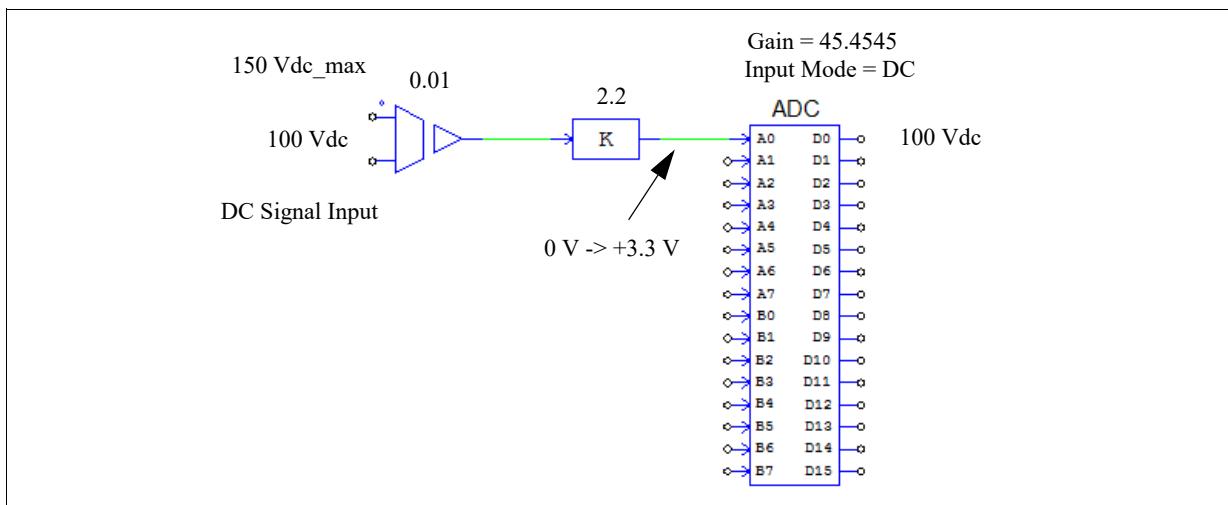
$$V_{i_s_c} = 1 * 2.2 = 2.2 \text{ V}$$

The scaling block after the DSP A/D can be selected such that the original power circuit quantity is restored. In this example, a gain of 45.4545 will be used. Note that this is the reciprocal of the combined gain of the voltage sensor and the conditioning circuit. At the A/D output, the maximum value and the actual value are:

$$V_o = 45.4545 * 3.3 = 150 \text{ V}$$

$$V_o = 45.4545 * 2.2 = 100 \text{ V}$$

The gain of the A/D channel will be set to 45.4545. The circuit connection and the settings are shown below.



Please note that, in this example, if the gain of the proportional block is changed from 2.2 to 1.1, and the A/D gain is changed from 45.4545 to 90.909, the simulation results will be the same. But the generated hardware code will not be correct. This is because the hardware code assumes that the maximum input value is scaled to +3.3V, but in this case it is only +1.5V. Therefore, one must set up the circuit such that, in the dc mode, the maximum input value is scaled to be +3.3V.

In another example, assume that a power circuit voltage is an ac quantity, and the range is as follows:

$$V_i_{max} = +/- 75 \text{ V}$$

The input mode of the A/D converter will be set to ac, and the input range is from -1.65V to +1.65V. Assume that the actual value of the voltage has a peak value of:

$$V_i = +/- 50 \text{ V}$$

Let the voltage sensor gain be 0.01. After the voltage sensor, the maximum value and the actual value of the input become:

$$V_{i_max_s} = +/- 0.75 \text{ V}$$

$$V_{i_s} = +/- 0.5 \text{ V}$$

Since the A/D converter input range is from -1.65V to +1.65V, this signal must be scaled before it is sent to the DSP. A conditioning circuit with a gain of 2.2 is needed (i.e. $1.65/0.75 = 2.2$). After the conditioning circuit and at the input of the DSP A/D converter, the maximum value and the actual value of the input become:

$$V_{i_max_s_c} = +/- 1.65 \text{ V}$$

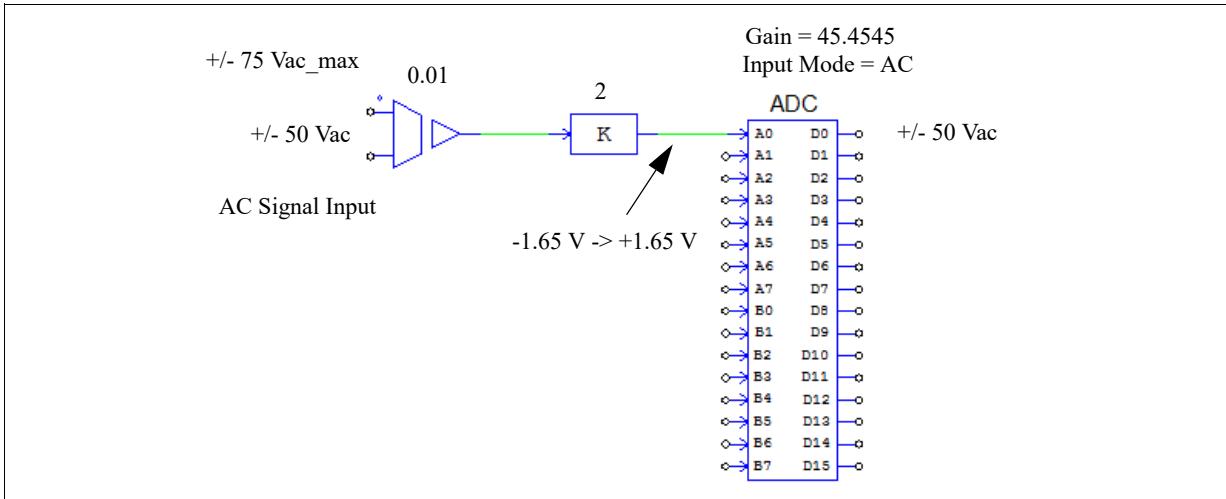
$$V_{i_s_c} = +/- 1.1 \text{ V}$$

The scaling block after the DSP A/D can be selected such that the original power circuit quantity is restored. In this example, a gain of 45.4545 will be used. Note that this is the reciprocal of the combined gain of the voltage sensor and the conditioning circuit. At the A/D output, the maximum value and the actual value are:

$$V_{o_max} = +/- 75 \text{ V}$$

$$V_o = +/- 50 \text{ V}$$

The gain of the A/D channel in PSIM will be set to 45.4545. The circuit connection and the settings are shown in the figure below.



Notice that in this circuit, the ac signal is sent to the A/D converter directly. This is because that, when the A/D input mode is set to AC, the input range is from -1.65V and +1.65V, and the function of the conditioning circuit that performs the dc offset is already included in the A/D converter block. In the actual hardware circuit, the ac signal must be scaled and offset so that the range is within 0V to +3.3V required by the A/D converter.

8.9 Comparator

F2806x supports three comparator modules. Each comparator block can have two external analog inputs, or one external analog input and one internal DAC reference for the other input. The comparator output can be sent to the PWM trip-zone and to the GPIO output.

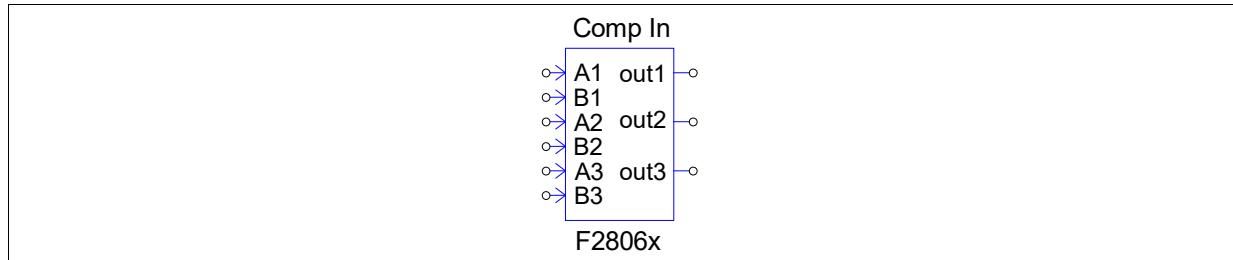
8.9.1 Comparator Input

In F2806x, there are 3 comparators. The 3 pairs of inputs share the same ADC/AIO ports with ADC input channels as shown below:

- Comparator 1 input A - Port ADCA2/AIO2
- Comparator 1 input B - Port ADCB2/AIO10
- Comparator 2 input A - Port ADCA4/AIO4
- Comparator 2 input B - Port ADCB4/AIO12
- Comparator 3 input A - Port ADCA6/AIO6
- Comparator 3 input B - Port ADCB6/AIO14

Only one function can be designated for each port. Simcoder will report error if a port is defined as comparator input but is used as a A/D converter channel or AIO in the same PSIM circuit schematic.

Image:



Attributes

Parameters	Description
Comparator i	Define how the output of the i_{th} comparator is used. It can be one of the following: <ul style="list-style-type: none"> - <i>Do not use</i>: Not used - <i>As a normal comparator</i>: Used as a normal comparator - <i>As a Trip-Zone signal</i>: Used as a trip-zone signal
Output Logic	Define the comparator output logic. It can be: <ul style="list-style-type: none"> - <i>High when $A > B$</i>: The output is high when Input A is greater than B. - <i>High when $A < B$</i>: The output is high when Input A is less than B.
Compare Method	Define how Input A of the comparator is compared to Input B. it can be one of the following: <ul style="list-style-type: none"> - <i>Compare to Input B</i>: Input A is compared to Input B where Input B is an external analog signal. - <i>Compare to Constant Value</i>: Input A is compared to a constant value. - <i>Compare to DAC output</i>: Input A is compared to a DAC output which is an internal signal.
Constant Value	The constant value when <i>Compare Method</i> is defined as <i>Compare to Constant Value</i> . The range of the constant value is from 0 to 3.3V.

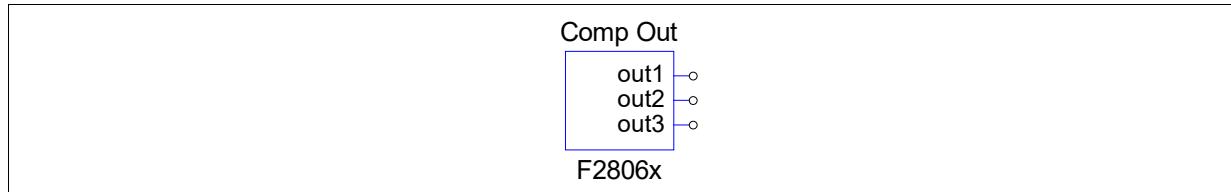
For all comparators, Input A is always from an external analog input. If Input B is also from another external analog input (if the parameter *Compare Method* is defined as *Compare to Input B*), the corresponding port must be defined as a comparator input in the Hardware Configuration block.

If the compare method is to compare to a constant value, Input B needs to be connected to ground in the schematic. If the compare method is to compare to a DAC output, Input B needs to be connected a comparator DAC output. In both cases, the corresponding ADC/AIO port can be used for other functions.

8.9.2 Comparator Output

The output of the comparator can be used as PWM trip-zone signal, as well as the GPIO output.

Image:



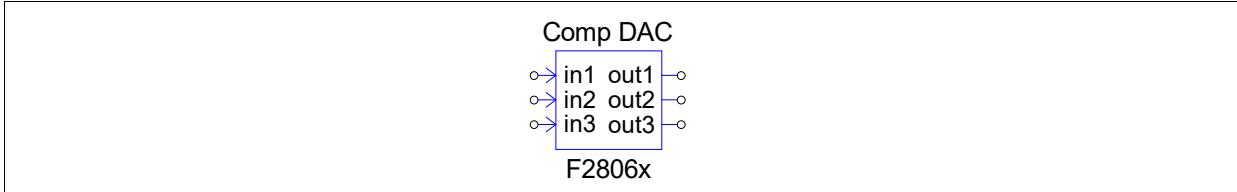
Attributes:

Parameters	Description
Comparator i Output	Output port position of the i_{th} comparator. It can be: <ul style="list-style-type: none"> - For Comparator A: GPIO1, 20, or 42 - For Comparator B: GPIO3, 21, 34, or 43 - For Comparator C: GPIO34

A comparator output block must be used together with a comparator input block. When a comparator output channel is used, the corresponding comparator input channel must be defined.

8.9.3 Comparator DAC

Each comparator block contains a 10-bit DAC reference that can be used by the inverting input (Input B) of the comparator.

Image:**Attributes:**

Parameters	Description
DAC i Range	The upper limit of the input signal range of the i_{th} comparator DAC. The lower limit is 0.
Use Ramp Generator	Define if the ramp generator is used in the comparator DAC.
Total Ramp Compensation	The total compensation of the ramp generator in one PWM period. It represents the total decrease of the ramp in one cycle.

Outputs of the comparator DAC can only be connected to the corresponding inverting inputs (Input B) of the comparator in a comparator input block. That is, node out1 can only be connected to node B1 of the comparator input block, and node out2 to node B2, and node out3 to B3. Also, a comparator DAC block cannot be used alone. It must be used in conjunction with a comparator input block.

If the ramp generator is not used, the input value is applied directly to DAC output immediately. The output range is from 0 to 3.3V, and the output can be calculated as follows:

$$\text{DAC Output} = \text{DAC Input} * 3.3 / \text{DAC Range}$$

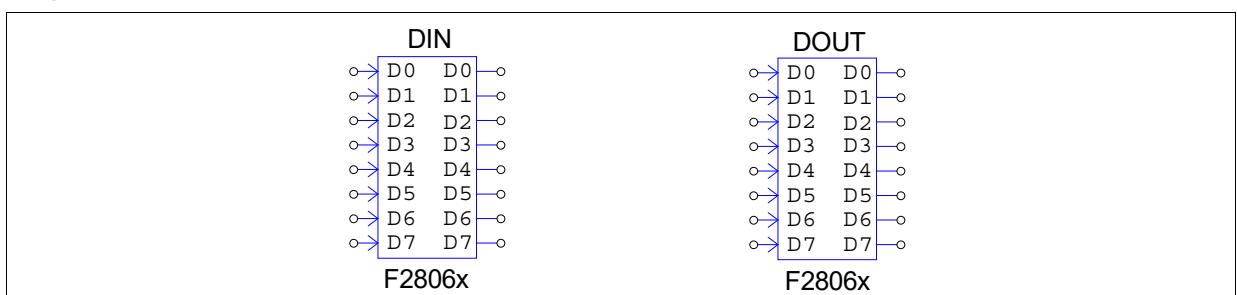
If the ramp generator is used, the input value is saved and used as the initial output value in the next PWM period. The comparator DAC output decreases linearly within the PWM period, and the total decrease is equal to the total ramp compensation value.

When the ramp generator is used, the sampling rate associated with the comparator DAC input must be the same as the frequency of the PWM generator that uses the comparator.

8.10 Digital Input and Digital Output

F2806x support 45 general-purpose-input-output (GPIO 0 to 44) ports that can be configured as either digital inputs or digital outputs. In addition, there are 6 digital analog-input-output ports (AIO 2, 4, 6, 10, 12, and 14) which also can be used as digital inputs and outputs.

In SimCoder, the digital inputs and outputs are grouped in 8-channel blocks. Multiple 8-channel digital input/output blocks can be used in the same schematic.

Image:

Attributes for Digital Input:

Parameters	Description
Port Position for Input i	The port position of the Input i , where i is from 0 to 7. It can be one of the 45 GPIO ports or one of the 6 AIO ports.
Use as External Interrupt	Indicate if this port is used as an external interrupt input.

Attributes for Digital Output:

Parameters	Description
Port Position for Output i	The port position of the Output i , where i is from 0 to 7. It can be one of the 45 GPIO ports or one of the 6 AIO ports.

Note that each GPIO and AIO port can be used for one function only. If an IO port is used as a digital input port, it can not be used as a digital output or any other peripheral port. For example, if Port GPIO1 is assigned as digital input and it is also used as PWM1 output, an error will be reported.

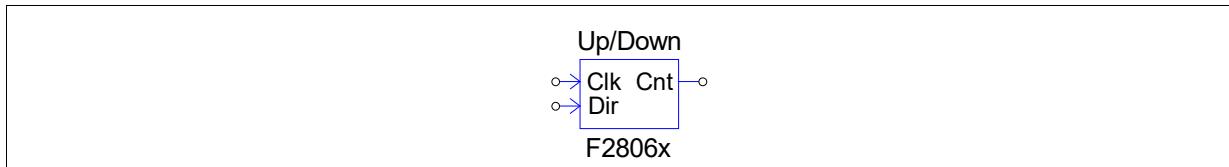
F2806x supports 3 masked external interrupts (XINT1 to XINT3). There are no dedicated pins for the external interrupts. XINT1, XINT2, and XINT3 interrupts can accept inputs from GPIO0 to GPIO31 pins.

8.11 Up/Down Counter

F2806x supports one up/down counter. The input ports are:

- GPIO20 for clock
- GPIO21 for direction

Image:



Attributes:

In the image, "Clk" refers to the input clock signal, and "Dir" refers to the signal that defines the counting direction. When the "Dir" input is 1, the counter counts forward, and when the input is 0, the counter counts backward.

The output of the up/down counter gives the counter value.

Note that the up/down counter uses the same resource as the encoder, and the same GPIO ports cannot be used in a counter and in an encoder at the same time. For example, using both Encoder 1 and Up/Down Counter 1 will cause conflict and is not allowed.

8.12 Encoder and Encoder State

F2806x supports an **Encoder** module. The GPIO ports used by encoder are:

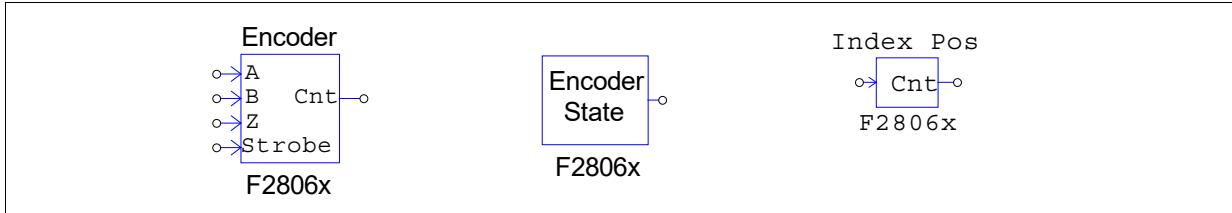
- GPIO20 for clock input
- GPIO21 for direction input
- GPIO22 for Z (or index signal) input
- GPIO23 for strobe signal input

The **Encoder State block** is used to indicate which input signal (either index signal or strobe signal) generates the interrupt. Also, hardware interrupt can be generated by the Z (index) signal and the strobe signal, and the output of the encoder state indicates which signal generates the interrupt. When the output is 0, the index signal generates the interrupt. When the output is 1, the strobe signal generates the interrupt.

The **Encoder Index/Strobe Position block** is used to latch the encoder's initial position. When the input of this

block is 0, the encoder counter is set to 0. Encoder will start to act when the input changes to 1. Encoder will latch the counter value when it meet the index/strobe event.

Images:



Attributes for Encoder:

Parameters	Description
Use Z Signal	Define if the encoder uses the Z (or index) signal.
Use Strobe Signal	Define if the encoder uses the strobe signal.
Counting Direction	The counting direction - Forward: the encoder counts up. - Reverse: the encoder counts down.
Z Signal Polarity	Define the trigger polarity of Z signal. - Active High - Active Low.
Strobe Signal Polarity	Define the trigger polarity of strobe signal. - Active High - Active Low.
Encoder Resolution	The resolution of the external encoder hardware. If it is 0, the encoder counter will keep on counting and will not reset. If for example, the resolution is set to 4096, the counter will be reset to 0 after it reaches 4095.

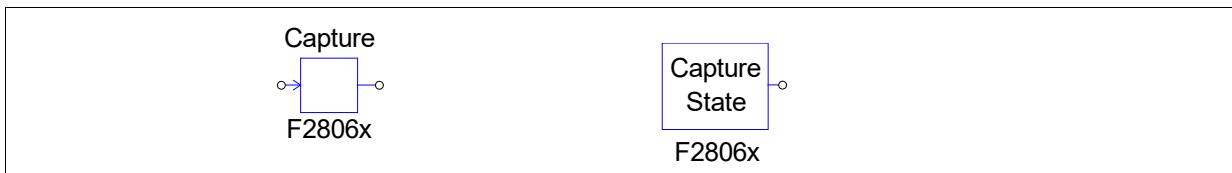
Attributes for Encoder Index/Strobe Pos:

Parameters	Description
Latch Position	Specify the kept counter type, choose from the followings: - IndexPos, if the setting "Use Z Signal" is not "No" in Encoder - StrobePos, if the setting "Use Strobe Signal" is not "No" in Encoder
Type of Position	This can be chosen from the followings: - The first latched position, or - The current latched position

8.13 Capture and Capture State

F2806x contains an enhanced capture module. A capture can generate interrupt, and the interrupt trigger mode is defined by the interrupt block.

Image:



Attributes for Capture:

Parameters	Description
Capture Source	Source of the capture may come from one of 3 GPIO ports, as listed below: - Capture 1 (GPIO5) - Capture 1 (GPIO19) - Capture 1 (GPIO24)
Event Filter Prescale	Event filter prescale. The input signal is divided by the selected prescale.
Timer Mode	Capture counter timer mode. It can be either <i>Absolute time</i> or <i>Time difference</i> .

Attributes for Capture State:

Parameters	Description
Capture Source	Source of the capture. It has only one source <i>Capture1</i> .

The Capture State block output is either 1 or 0, where 1 means the rising edge and 0 means the falling edge.

8.14 Serial Communication Interface (SCI)

F2806x provides the function for serial communication interface (SCI). Through SCI, data inside the DSP can be transferred to a computer using an external RS-232 cable. PSIM provides all the necessary functions to transmit and receive data on both the DSP and computer sides, and to display the data on the computer. This provides a very convenient way to monitor, debug, and adjust the DSP code in real time.

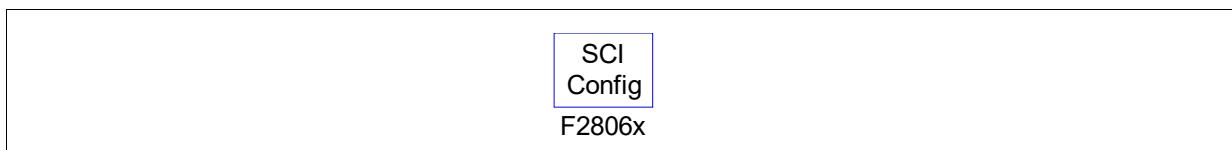
For more detailed descriptions on SCI and the monitoring function, please refer to the document "Tutorial - Using SCI for Real-Time Monitoring.pdf".

Three SCI function blocks are provided in SimCoder: *SCI Configuration*, *SCI Input*, and *SCI Output*, as described below.

8.14.1 SCI Configuration

The SCI Configuration block defines the SCI port, the communication speed, the parity check type, and the data buffer size.

Image:



Attributes:

Parameters	Description
SCI Port	Define the SCI port. Different sets of GPIO ports can be used for SCI, as listed below: - SCIA: GPIO28 and 7 in combination with GPIO29 and 12 - SCIB: GPIO11, 15, 19, 23, 41, and 44 in combination with GPIO9, 14, 18, 22, 40, and 58
Speed (bps)	SCI communication speed, in bps (bits per second). A list of preset speeds is provided at 200000, 115200, 57600, 38400, 19200, or 9600 bps. Or one can specify any other speed manually.

Parity Check	The parity check setting for error check in communication. It can be either <i>None</i> , <i>Odd</i> , or <i>Even</i> .
Output Buffer Size	Size of the data buffer allocated in DSP for SCI. The buffer is located in the RAM area, and each buffer cell stores one data point which consists of three 16-bit words (that is, 6 bytes, or 48 bits, per data point).

Note that the buffer size should be properly selected. On one hand, a large buffer is preferred in order to collect more data points so that more variables can be monitored over a longer period of time. On the other hand, the internal DSP memory is limited, and the buffer should not be too large to interfere with the normal DSP operation.

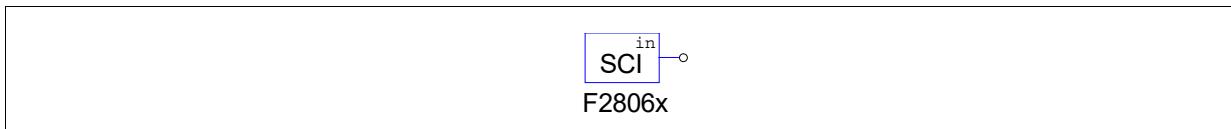
For more information on how to select the buffer size, please refer to the document "*Tutorial - Using SCI for Real-Time Monitoring.pdf*".

8.14.2 SCI Input

The SCI Input block is used to define a variable in the DSP code that can be changed. The name of the SCI input variable will appear in the DSP Oscilloscope (under the **Utilities** menu), and the value can be changed at runtime via SCI.

The SCI input block provides a convenient way to change reference, or fine tune controller parameters, for example.

Image:



Attributes:

Parameters	Description
Initial Value	The initial value of the SCI input variable.

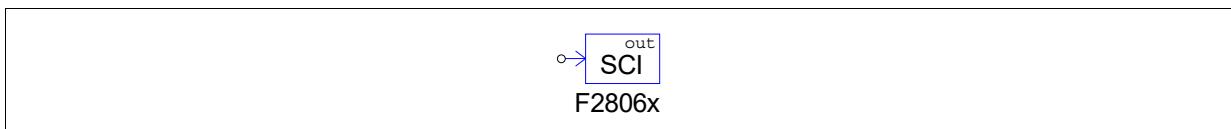
In the schematic, the SCI input behaves as a constant. While its value can be changed at runtime when the code is running on the DSP, the value will be fixed at the initial value in the simulation.

8.14.3 SCI Output

The SCI Output block is used to define a variable for display. When a SCI output block is connected to a node, the name of the SCI output block will appear in the DSP Oscilloscope (under the **Utilities** menu), and data of this variable can be transmitted from DSP to the computer via SCI at runtime, and the waveform can be displayed in the DSP Oscilloscope.

The SCI output block provides a convenient way to monitor DSP waveforms.

Image:



Attributes:

Parameters	Description
Data Point Step	It defines how frequent data is collected. If the Data Point Step is 1, every data point is collected and transmitted. If the Data Point Step is 10, for example, only one point of out every 10 points is collected and transmitted.

Note that if the Data Point Step is too small, there may be too many data points and it may not be possible to transmit them all. In this case, some data points will be discarded during the data transmission.

Also, the Data Point Step parameter is used only when the DSP Oscilloscope is in the *continuous* mode. When it is in the *snapshot* node, this parameter is ignored and every point is collected and transmitted.

In simulation, the SCI output behaves as a voltage probe.

8.15 Serial Peripheral Interface (SPI)

F2806x provides the functions for serial peripheral interface (SPI). By using the SPI blocks in the TI F803x Target library, one can implement the function to communicate with external SPI devices (such as external A/D and D/A converters) easily and conveniently. Writing code manually for SPI devices is often a time-consuming and non-trivial task. With the capability to support SPI, PSIM greatly simplifies and speeds up the coding and hardware implementation process.

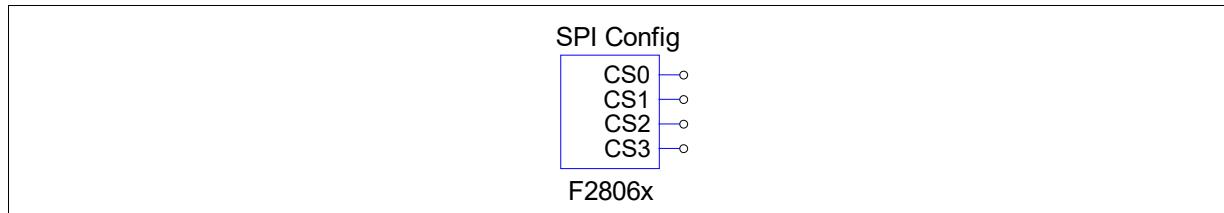
For more detailed descriptions on how to use SPI blocks, please refer to the document "*Tutorial - Using SPI for Real-Time Monitoring.pdf*".

Four SPI function blocks are provided in SimCoder: *SPI Configuration*, *SPI Device*, *SPI Input*, and *SPI Output*, as described below.

8.15.1 SPI Configuration

The SPI Configuration block defines the SPI port, the chip selection pins, and the SPI buffer size. It must be present in a schematic where SPI is used, and this block must be in the main schematic.

Image:



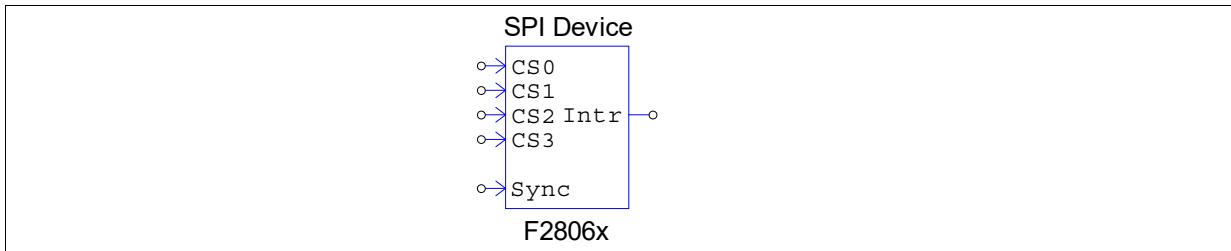
Attributes:

Parameters	Description
SPI Port	Define the SPI port from the options: <ul style="list-style-type: none">- SPIA (GPIO 16-19)- SPIA (GPIO 3, 5, 18, 19)- SPIB (GPIO 12-15)- SPIB (GPIO 24-27)
Chip Select Pin0, 1, 2, and 3	The GPIO port of the chip select pin. PSIM supports up to 16 SPI devices, which requires four GPIO pins for chip select, as defined by Chip Select Pin0 to Pin3. These GPIO ports and the SPI slave transmit-enable pin SPISTE are used to generate the chip select signal.
SPI Buffer Size	The buffer size of the SPI commands. Each memory cell of the buffer saves the index of a SPI command. Normally, one can specify the buffer size as 1 plus the number of SPI commands (i.e. Start Conversion Command, Receiving Data Command, Sending Data Command, and Sync. Command) in all SPI Input/Output elements.

8.15.2 SPI Device

The SPI Device block defines the information of the corresponding SPI hardware device. The number of SPI Device blocks in the schematic must be the same as the number of SPI hardware devices.

Image:



Attributes:

Parameters	Description
Chip Select Pins	The state of the chip select pins corresponding to the SPI device. When the chip select pins are at this state, this SPI device is selected.
Communication Speed (MHz)	SPI communication speed, in MHz.
Clock Type	SPI clock type, as determined by the SPI hardware device. It can be one of the following: - <i>Rising edge without delay</i> : The clock is normally low, and data is latched at the clock rising edge. - <i>Rising edge with delay</i> : The clock is normally low, and data is latched at the clock rising edge with delay. - <i>Falling edge without delay</i> : The clock is normally high, and data is latched at the clock falling edge. - <i>Falling edge with delay</i> : The clock is normally high, and data is latched at the clock falling edge with delay.
Command Word Length	Word length, or the length of the significant bits, of SPI communication commands. It can be from 1 to 16 bits.
Sync. Active Mode	The triggering mode of the synchronization signal of the SPI device. It can be either <i>Rising edge</i> or <i>Falling edge</i> .
SPI Initial Command	The SPI command that initializes the SPI device.
Hardware Interrupt Mode	Specify the type of the interrupt signal that the SPI device generates. This is valid only when the SPI device's interrupt output node is connected to the input of a digital output element. It can be one of the following: - <i>No hardware interrupt</i> - <i>Rising edge</i> - <i>Falling edge</i>

Interrupt Timing	<p>Specify how a SPI device generates interrupt when it completes conversion. It can be one of the following:</p> <ul style="list-style-type: none"> - <i>No interrupt</i>: No interrupt is generated. In this case, DSP sends the command to a SPI input device. This device starts the conversion and returns the result in the same command - <i>Multiple interrupt in series</i>: Multiple interrupts are generated in series after each conversion. This is for a SPI device that has one A/D conversion unit and multiple input channels. In this case, DSP send the first conversion command, and the SPI device starts the conversion. When the conversion is complete, the SPI device will generate an interrupt. In the interrupt service routine, DSP will send a command to fetch the conversion result, and start a new conversion of another channel of the same SPI input device. - <i>One-time interrupt</i>: Only one interrupt is generated at the end of the conversion. This is for a SPI device that can perform multiple channel conversions in one request. In this case, DSP sends the command to the SPI input device, and the SPI device completes the conversion of multiple input channels. When all the conversions are complete, the SPI device will generate an interrupt.
Command Gaps (ns)	The gap between two SPI commands, in nsec.
Conversion Sequence	Define the names of the SPI input elements, separated by comma, that determine the conversion sequence. Note that this parameter is valid only when the SPI device generates multiple interrupts in series.

In a schematic, the chip select pins of all the SPI devices are connected to the chip select pins of the SPI Configuration block, without defining how the chip select logic is implemented. In the actual hardware, however, one would need to implement the corresponding chip select logic accordingly.

A SPI command consists of a series of 16-bit numbers separated by comma. In the 16-bit number, only the lower bits are the significant bits used by the command. For example, if the Command Word Length is 8, Bits 0 to 7 are the command, and Bits 8 to 15 are not used.

A SPI device can be either an input device or an output device. For example, an external A/D converter is an input device. Usually DSP will send one or multiple A/D conversion commands to the device, and then set the synchronization signal to start the conversion. The synchronization signal is reset at the next command of the same device.

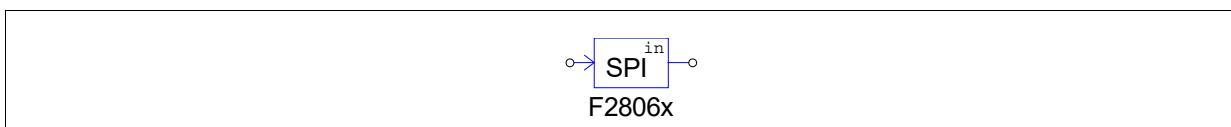
A SPI input device using the synchronization signal usually needs an interrupt pin to trigger DSP to enter the interrupt service routine.

On the other hand, an external D/A converter is an output device. Usually DSP sends one or multiple D/A conversion commands to the device, and then sets the synchronization signal to start the conversion. The synchronization signal is reset at the next command of the same device.

8.15.3 SPI Input

A SPI input device may have multiple input channels. The SPI Input block is used to define the properties of an input channel for SPI communication, and one SPI Input block corresponds to one input channel.

Image:



Attributes:

Parameters	Description
Device Name	Name of the SPI input device.

Start Conversion Command	Command to start conversion, in hex numbers, separated by comma (for example, 0x23,0x43,0x00).
Receiving Data Command	Command to receive data, in hex numbers, separated by comma (for example, 0x23,0x43,0x00).
Data Bit Position	Define where the data bits are in the receiving data string. The format is: $ElementName = \{Xn[MSB..LSB]\}$ where <ul style="list-style-type: none"> - $ElementName$ is the name of the SPI input device. If it is the current SPI input device, use y instead. - $\{\}$ means that the item in the bracket repeats multiple times. - Xn is the n_{th} word received from the SPI input device, and n start from 0. - $MSB..LSB$ defines the position of the significant bits in the word.
Input Range	Specify the parameter V_{max} that defines the input range. This parameter is valid only when the SPI device is an A/D converter. If the device conversion mode is DC, the input ranges from 0 to V_{max} . If the device conversion mode is AC, the input ranges from $-V_{max}/2$ to $V_{max}/2$.
Scale Factor	Output scale factor K_{scale} . If the scale factor is 0, the SPI device is not an A/D converter, and the result will be exactly the same as what DSP receives from SPI communication. Otherwise, the SPI device is an A/D converter, and the result is scaled based on this factor and the A/D conversion mode.
ADC Mode	The A/D conversion mode of the device. It can be either DC or AC. Note that this parameter is valid only when the device is an A/D converter.
Initial Value	The initial value of the input.

The formula for the *Data Bit Position* defines the data length of a SPI input device. For example, $y=x1[3..0]x2[7..0]$, means that the data length is 12, and the result is the lower 4 bits of the 2nd word and the lower 8 bits of the 3rd word. If the received data string is 0x12,0x78,0xAF, then the result is 0x8AF.

If the scale factor is not 0, the output will be scaled based on the following:

In the DC conversion mode:

- In simulation: $Output = Input \cdot K_{scale}$
- In hardware: $Output = \frac{Result \cdot V_{max} \cdot K_{scale}}{2^{Data_Length}}$

In the AC conversion mode:

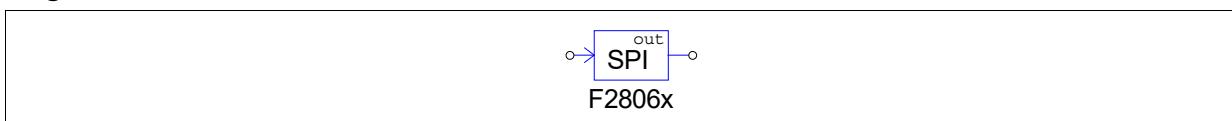
- In simulation: $Output = Input \cdot K_{scale}$
- In hardware: $Output = \frac{(Result - 2^{Data_Length-1}) \cdot V_{max} \cdot K_{scale}}{2^{Data_Length-1}}$

The parameter *Data Length* is calculated from the Data Bit Position formula.

8.15.4 SPI Output

A SPI output device may have multiple output channels. The SPI Output block is used to define the properties of an output channel for SPI communication, and one SPI Output block corresponds to one output channel.

Image:



Attributes:

Parameters	Description
Device Name	Name of the SPI output device.
Scale Factor	Output scale factor K_{scale} . If the scale factor is 0, the SPI device is not a D/A converter, and the result will be exactly the same as what DSP receives from SPI communication. Otherwise, the SPI device is an D/A converter, and the result is scaled based on this factor and the D/A conversion mode.
Output Range	Specify the parameter V_{max} that defines the output range. This parameter is valid only when the SPI device is an D/A converter. If the device conversion mode is DC, the input ranges from 0 to V_{max} . If the device conversion mode is AC, the input ranges from $-V_{max}/2$ to $V_{max}/2$.
DAC Mode	The D/A conversion mode of the device. It can be either <i>DC</i> or <i>AC</i> . Note that this parameter is valid only when the device is a D/A converter.
Sending Data Command	Command to send the output data, in hex numbers, separated by comma (for example, 0x23,0x43,0x00).
Data Bit Position	Define where the data bits are in the sending data string. The format is: $ElementName = \{Xn[MSB..LSB]\}$ <p>where</p> <ul style="list-style-type: none"> - $ElementName$ is the name of the SPI output device. If it is the current SPI output device, use y instead. - $\{\}$ means that the item in the bracket repeats multiple times. - Xn is the n_{th} word sent to the SPI output device, and n start from 0. - $MSB..LSB$ defines the position of the significant bits in the word.
Sync. Command	The command to synchronize output channels of the SPI output device, in hex numbers, separated by comma (for example, 0x23,0x43,0x00). This command is used when the SPI output device does not have the synchronization signal

The formula for the *Data Bit Position* defines the data length of a SPI output device. For example, $y=x1[3..0]x2[7..0]$, means that the data length is 12, and the data is the lower 4 bits of the 2nd word and the lower 8 bits of the 3rd word. If the sending data string is 0x12,0x78,0xAF, then the data is 0x8AF.

If the scale factor is not 0, the output will be scaled based on the following:

In the DC conversion mode:

$$\begin{aligned} \text{- In simulation: } & Output = Input \cdot K_{scale} \\ \text{- In hardware: } & Output = \frac{Result \cdot K_{scale} \cdot 2^{Data_Length}}{V_{max}} \end{aligned}$$

In the AC conversion mode:

$$\begin{aligned} \text{- In simulation: } & Output = Input \cdot K_{scale} \\ \text{- In hardware: } & Output = 2^{Data_Length} + \frac{Result \cdot K_{scale} \cdot 2^{Data_Length-1}}{V_{max}} \end{aligned}$$

The parameter *Data_Length* is calculated from the Data Bit Position formula.

8.16 Controller Area Network (CAN) Bus

F2833x provides the function for CAN bus communication. PSIM provides the necessary functions to implement CAN bus.

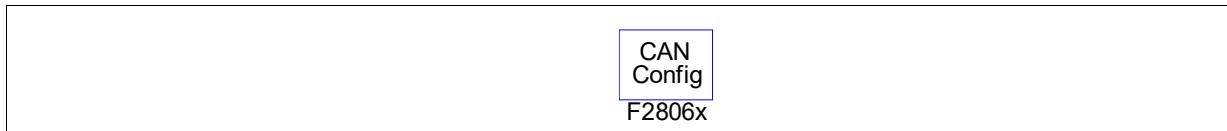
Three function blocks are provided in SimCoder: *CAN Configuration*, *CAN Input*, and *CAN Output*, as described below.

8.16.1 CAN Configuration

The CAN Configuration block defines the CAN bus source, data byte order, and other settings.

F2806x DSP has one CAN source: CAN A with GPIO 31 for transmit and GPIO 30 for receive.

Image:



Attributes:

Parameters	Description
CAN Speed	Communication speed, in Hz. It can be set to one of the following preset values: 125kHz, 250kHz, 500kHz, and 1MHz Or you can set the speed manually by typing the text in the parameter field. Note that the speed should not exceed 1MHz.
Data Byte Order	The order of the data bytes. It can be one of the following: - Least significant byte 1st - Most significant byte 1st "Least significant byte 1st" means that the least significant byte is placed first; while "Most significant byte 1st" means that the most significant byte is placed first.
Number of Input Mailboxes	Number of input mailboxes
Checking Receive Mail Lost	If it is set to <i>Enable</i> , the error report function " <code>_ProcCanAErrReport(nErr)</code> " (for CAN A) will be called if the received mail is lost. This function will return the value of the error status nErr from the CAN register CANGIF0. If it is set to <i>Disable</i> , the error report function will not be called.
Checking Bus Off	If it is set to <i>Enable</i> , the error report function " <code>_ProcCanAErrReport(nErr)</code> " (for CAN A) will be called if the bus is in the off state. This function will return the value of the error status nErr from the CAN register CANGIF0. If it is set to <i>Disable</i> , the error report function will not be called.
Error Check Mode	The error check mode can be either <i>Passive</i> or <i>Active</i> . If it is in the error-passive mode, an interrupt will be generated when the error count reaches 128. If it is in the error-active mode, an interrupt will be generated every time.

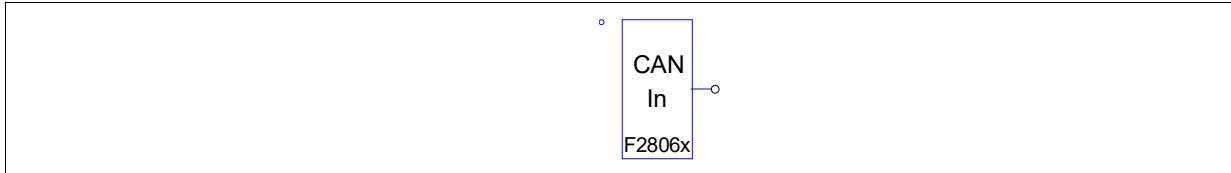
The returned status nErr in the function "`_ProcCanAErrReport(nErr)`" (for CAN A) is a 32-bit integer. It obtains its value from the Global Interrupt Flag Register CANGIF0. After returning from the function "`_ProcCanAErrReport(nErr)`", the register CANGIF0 will be cleared.

Also, if you wish to take actions on a specific error, you can add your own code within the "`_ProcCanAErrReport(nErr)`" function.

8.16.2 CAN Input

A CAN Input block receives CAN messages from a CAN bus.

Image:



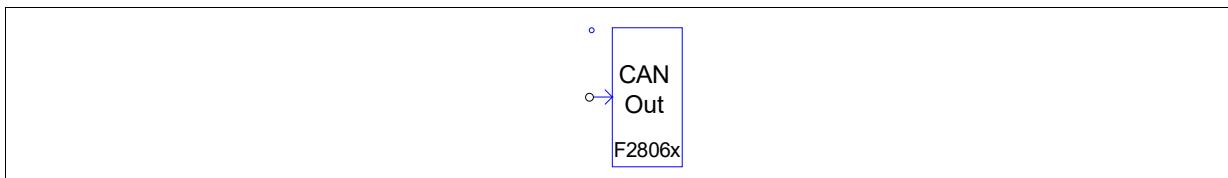
Attributes:

Parameters	Description
Number of Inputs	Number of CAN inputs. It can have up to 8 inputs.
Use Extension ID	If this is set to <i>Yes</i> , the ID of a message is a 29-bit integer. If this is set to <i>No</i> , the ID of a message is a 11-bit integer.
Message ID	The ID of a message. It is an integer, for example, 0x23.
Local Mask	The mask for the message. It is an integer. If the bits of the message ID matches the bits of the mask, the message will be received. Otherwise, it will be ignored. For example, if the mask is 0x380 and the message ID is 0x389, this message will be received. But if the message ID is 0x480, the message will be ignored.
Overwrite Flag	It can be set to <i>Allow overwrite</i> or <i>Do not allow overwrite</i> . Assume a mailbox is configured to accept a message, and there is a new message coming from the CAN bus, while the old message is still in the mailbox and has not been processed yet. If the flag is set to "Allow overwrite", the new message will be accepted, and the old message will be overwritten. If the flag is set to "Do not allow overwrite", the new message will not be accepted, and the old one will be kept.
Receive Message Rate	The number of messages received by the input block in a specific period of time. For example, there are two input blocks, with the first input block receiving 20 messages per second, and the second input block receiving 30 messages per second. The parameter "Receive Message Rate" will be set to 20 for the first block, and 30 for the second block.
Input i Gain	The gain to the i_{th} input where i can be 1 to 8. The output is the input multiplied by the gain.
Input i Data Start Position	A message can have up to 8 data points. A data point can have 1 bit up to 32 bits. This defines the start position of the current data point in the message.
Input i Data End Position	This defines the end position of the current data point in the message.
Input i Data Type	The data type can be either <i>Float</i> , <i>Integer</i> , or <i>IQ1</i> to <i>IQ30</i> .
Input i Default Data	The initial value of the SCI input variable.

8.16.3 CAN Output

A CAN Output block transmits CAN messages to a CAN bus.

Image:



Attributes:

Parameters	Description
Number of Outputs	Number of CAN outputs. It can have up to 8 outputs.
CAN Source	The CAN source can be either CAN A or CAN B.
Use Extension ID	If this is set to <i>Yes</i> , the ID of a message is a 29-bit integer. If this is set to <i>No</i> , the ID of a message is a 11-bit integer.
Message ID	The ID of a message. It is an integer, for example, 0x23.
Trigger Type	The trigger type can be one of the following: <ul style="list-style-type: none"> - <i>No trigger</i>: No triggering - <i>Rising edge</i>: Triggering occurs at the rising edge of the trigger source. - <i>Falling edge</i>: Triggering occurs at the falling edge of the trigger source. - <i>Rising/falling edge</i>: Triggering occurs at both the rising edge and the falling edge of the trigger source. <p>A rising/falling edge is considered to have occurred if the difference between the current value of the trigger source and the value at the last triggering instant is equal to or greater than 1.</p>
Trigger Source	A trigger source can be either a constant or a global variable depending on the Trigger Type. <p>If the Trigger Type is set to "No trigger", the trigger source defines the counter limit. For example, if the trigger source is a constant or a global value, and the value is 5, it means that triggering will occur once out of every 5 times. In another word, the data will be sent out once per every 5 cycles.</p> <p>If the Trigger Type is set to edge trigger, the trigger source can only be a global variable. Triggering will occur when the global variable has the rising or falling edge or both depending on the Trigger Type.</p>
Output <i>i</i> Gain	The gain to the i_{th} output where i can be 1 to 8. The output is the output multiplied by the gain.
Output <i>i</i> Data Start Position	A message can have up to 8 data points. A data point can have 1 bit up to 32 bits. This defines the start position of the current data point in the message.
Output <i>i</i> Data End Position	This defines the end position of the current data point in the message.
Output <i>i</i> Data Type	The data type can be either <i>Float</i> , <i>Integer</i> , or <i>IQ1</i> to <i>IQ30</i> .
Output <i>i</i> Default Data	The initial value of the SCI output variable.

8.17 Interrupt Time

The interrupt time block is used to measure the time interval of an interrupt service routine.

Attributes:

Parameters	Description
Time Output Method	Define how interrupt time is measured. It can be one of the following: - <i>SCI (time used)</i> : Using SCI. Time used by the interrupt service routine, in DSP clock count, is measured and is sent out via SCI output. - <i>SCI (time remaining)</i> : Using SCI. Time remaining in the interrupt service routine, in DSP clock count, is measured and is send out via SCI output. The time remaining is defined as the time from the end of the current interrupt to the beginning of the next interrupt. - <i>GPIO0 to GPIO44, or AIO2 to AIO14</i> : Using a GPIO port. A pulse is generated at the specified GPIO port. The pulse is set to high when entering the interrupt, and set to low when exiting the interrupt. An oscilloscope can be used to measure the width of the pulse.
Sampling Frequency	Sampling frequency of the interrupt service routine, in Hz.

When SCI is used, the value is the count of the DSP clock. For example, if the value is 6000, for a 60-MHz DSP clock, the interrupt time will be: $6000 / 60M = 100 \text{ us}$.

8.18 Project Settings and Memory Allocation

When generating the code for the TI F2806x Hardware Target, SimCoder also creates the complete project files for the TI Code Composer Studio (CCS) development environment, so that the code can be compiled, linked, and uploaded to the DSP.

At the present, CCS version 3.3 is supported. Assuming that the PSIM schematic file is "test.sch", after the code generation, a sub-folder called "test (C code)" will be generated in the directory of the schematic file, and sub-folder will contain the following files:

- test.c Generated C code
- PS_bios.h: Header file for the SimCoder F2806x library
- passwords.asm: File for specifying the DSP code password
- test.pjt: Project file for Code Composer Studio
- F2806x_Headers_nonBIOS.cmd: Peripheral register linker command file
- F28069_FLASH_Lnk.cmd: Flash memory linker command file
- F28069_FLASH_RAM_Lnk.cmd: Flash RAM memory linker command file
- F28069_RAM_Lnk.cmd: RAM memory linker command file

Note: The names of the link command files are assigned with the target hardware if it is not F2806x. For example, if the target hardware is F28069, the file names will be *F28069 FLASH Lnk.cmd*, *F28069 FLASH RAM Lnk.cmd*, and *F28069RAM Lnk.cmd* accordingly.

Besides, the project also needs the following library files:

- PsBiosRamF06xFixpt.lib: SimCoder F2806x library, located in the PSIM folder
- PsBiosRomF06xFixpt.lib: SimCoder F2806x library, located in the PSIM folder
- IQmath.lib: TI's IQmath.lib, located in the PSIM /lib folder
- 2806x_IQmath_BootROMsymbols.libIQmath symbols library, located in the PSIM /lib folder

These library files will be copied automatically to the project folder when the code is generated.

Each time code generation is performed, the .c file and .pjt file (in this example, "test.c" and "test.pjt") will be created. If you have made changes manually to these two files, be sure to copy the changed files to a different location. Otherwise the changes will be overwritten when code generation is performed next time.

Project Setting:

In the Code Composer Studio project file, the following settings are provided:

- *RAM Debug*: To compile the code in the debug mode and run it in the RAM memory
- *RAM Release*: To compile the code in the release mode and run it in the RAM memory
- *Flash Release*: To compile the code in the release mode and run it in the flash memory
- *Flash RAM Release*: To compile the code in the release mode and run it in the RAM memory

When RAM Debug or RAM Release is selected, CCS uses the linker command file F2806x_RAM_Lnk.cmd to allocate the program and data space.

When Flash Release is selected, CCS uses the linker command file F2806x_FLASH_Lnk.cmd to allocate the program and data space.

When Flash RAM Release is selected, CCS uses the linker command file F2806x_FLASH_RAM_Lnk.cmd to allocate the program and data space. The memory allocation is the same as in RAM Release.

The code compiled in the release mode is faster than the code in the debug mode. Also, the code in RAM Release or Flash RAM Release is the fastest. The code in RAM Debug is slower, and the code in Flash Release is the slowest. In a development, normally one would start with RAM Debug for easy debugging. Then switch to RAM Release and consequently to Flash Release or Flash RAM Release.

Memory Allocation:

In the generated link files, the memory allocation is defined in the following way.

With the RAM Debug, RAM Release, and Flash RAM Release settings:

RAM Memory 0x0000 - 0x07FF (2K) interrupt vectors stack 0x8000 - 0x13FFF (96K*) program and data space
--

With the Flash Release setting:

RAM Memory 0x0000 - 0x07FF (2K) interrupt vectors stack 0x8000 - 0x13FFF (96K*) data space	Flash Memory 0x3D8000 - 0x3F7FFF (256K**) program password etc.
--	--

Notes:

* The RAM memory predefined by SimCoder for program and data space is:

- For F28069, F28068, F28067, F28065, and F28064: from 0x8000 to 0x13FFF (96K)
- For F28066 and F28063: From 0x8000 to 0xFFFF (64K)
- For F28062: From 0x8000 to 0xDFFF (48K)
- If the combined program and data space exceeds the size of the RAM space, Flash Release must be selected as the project setting.

** The flash memory predefined by SimCoder for program space is:

- For F28069, F28068, F28067, and F28066: From 0x3E8000 to 0x3F7FFF (256K)
- For F28065, F28064, F28063, and F28062: From 0x3E8000 to 0x3F7FFF (128K)

F2802x Hardware Target

9.1 Overview

With the F2802x Hardware Target, SimCoder can generate code that is ready to run on any hardware boards based on Texas Instruments' F2802x fixed-point DSP.

The F2802x Hardware Target will work with all F2802x packages.

The F2802x Hardware Target library includes the following function blocks:

- PWM generators: 3-phase, 2-phase, 1-phase, and APWM
- Variable frequency PWM
- Start/Stop functions for PWM generators
- Trip-one and trip-zone state
- A/D converter
- Comparator input, output, and DAC
- Digital input and output
- Capture and capture state
- SCI configuration, Input, and output
- SPI configuration, device, input, and output
- CAN configuration, input, and output
- Interrupt Time
- DSP clock
- Hardware configuration

When generating the code for a system that has multiple sampling rates, SimCoder will use the interrupts of the PWM generators for the PWM sampling rates. For other sampling rates in the control system, it will use the Timer 1 interrupt first, and then Timer 2 interrupt if needed. If there are more than three sampling rates in the control system, the corresponding interrupt routines will be handled in the main program by software.

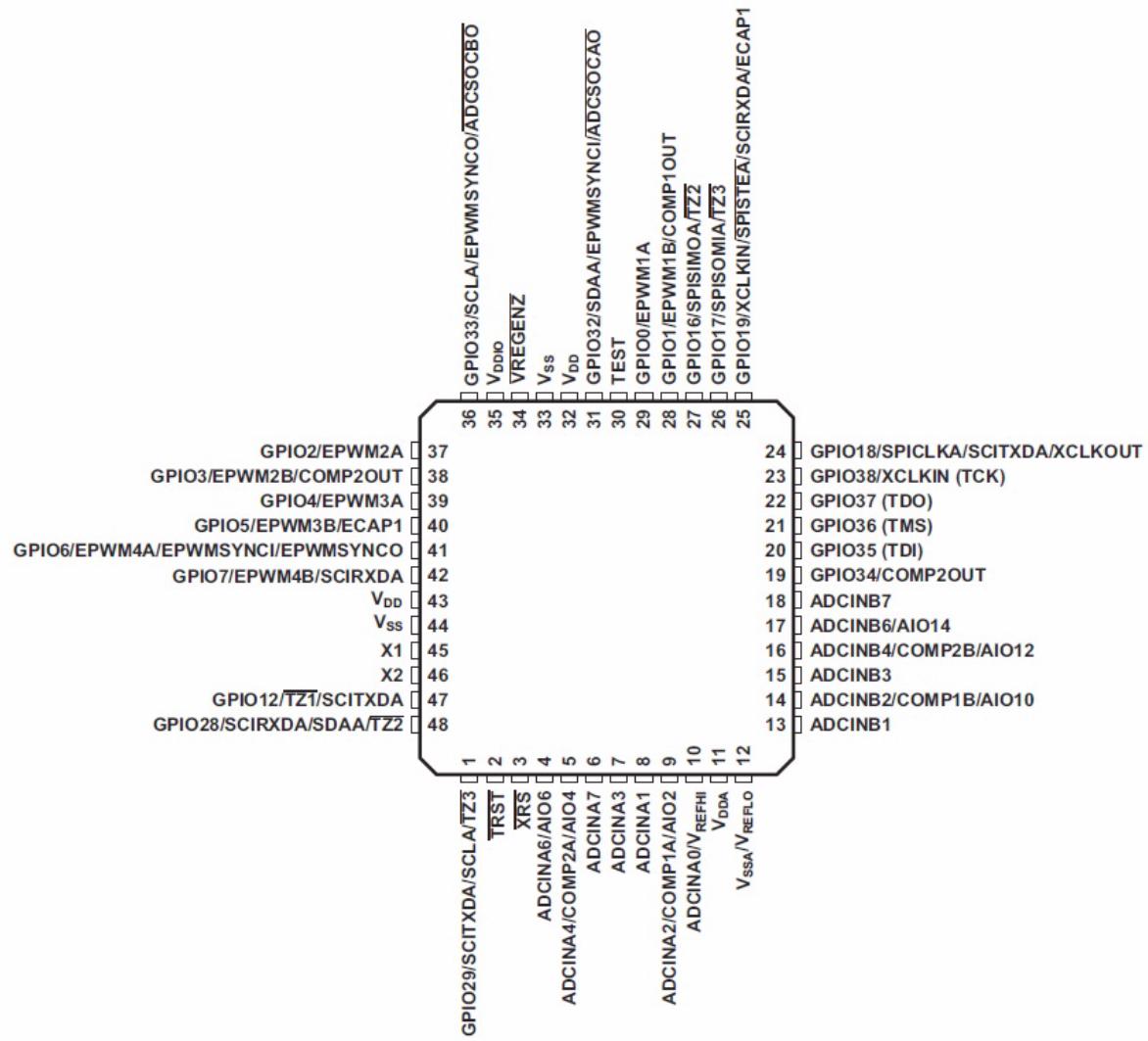
In TI F2802x, PWM generators can generate hardware interrupt. SimCoder will search and group all the elements that are connected to the PWM generator and have the same sampling rate as the PWM generator. These elements will be automatically placed and implemented in an interrupt service routine in the generated code.

In addition, digital input, encoder, capture, and trip-zone can also generate hardware interrupt. Each hardware interrupt must be associated with an interrupt block (described in Section 4.5 of this Manual), and each interrupt block must be associated with an interrupt service routine (a subcircuit that represents the interrupt service routine). For example, if a PWM generator and a digital input both generate interrupt, there should be one interrupt block and one interrupt service routine for each of them.

The definitions of the elements in the F2802x Hardware Target library are described in this Chapter.

The figure below shows the F2802x 48-pin PT LQFP port assignment.

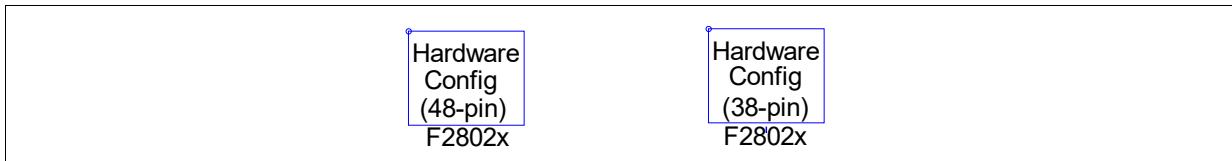
F2802x 48-Pin LQFP Port Assignment



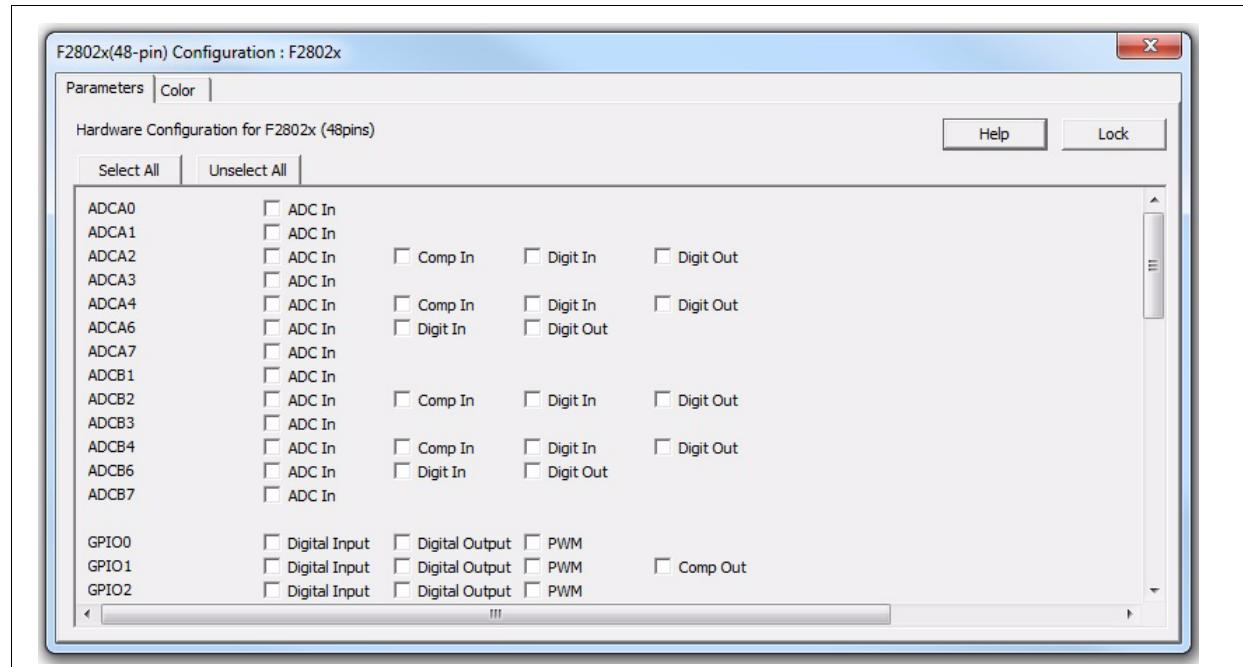
9.2 Hardware Configuration

F2802x provides a 16 channel analog inputs and up to 45 individually programmable multiplexed GPIO ports. Many of the GPIO ports can perform one of several functions. For example, port GPIO1 can be used either as a digital input, a digital output, or a PWM output. Some of the 16 A/D channels can also be defined as either digital input, digital output, or comparator input. Therefore, user must assign the functions to the GPIO ports correctly according to the PSIM circuit schematic.

Image:



The dialog window of the block is shown below:



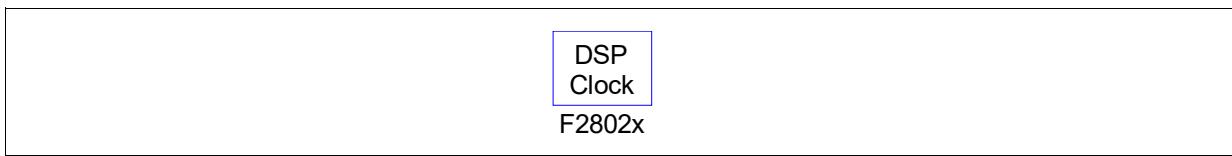
The Hardware Configuration block is for user to specify the I/O ports of the F2802x hardware. Every port to be used must be assigned correctly. The ports not in use can be left unchecked.

For each GPIO port, a check box is provided for each of its available function. When a box is checked, the GPIO port is configured for that particular function. For example, if the checkbox for "Digital Input" is checked for port GPIO1, this port is configured as a digital input, and hence, cannot be used for any other functions. If it is used as a PWM output in the PSIM circuit schematic, an error message will be generated.

9.3 DSP Clock

The DSP Configuration block defines the external clock frequency and the speed of the F2802x DSP, as well as the program space size.

Image:



Attributes:

Parameters	Description
DSP Clock Source	There are five ways of providing system clock to F2802x. They are as follows: <ul style="list-style-type: none">- Internal oscillator 1- Internal oscillator 2- External oscillator- External clock (GPIO19)- External clock (GPIO38)
External Clock (MHz)	Frequency of the external clock on the DSP board, in MHz. The frequency must be an integer, and the maximum frequency allowed is 10 MHz. This parameter is ignored if the DSP clock source is selected to be internal oscillator 1 or 2.
DSP Speed (MHz)	DSP Speed, in MHz. The speed must be an integer, and must be an integer multiple of the external clock frequency, from 1 to 12 times. The maximum DSP speed allowed is 60 MHz.

If a DSP Configuration block is not used in a circuit, the default values of the DSP Configuration block will be used.

9.4 PWM Generators

F2802x contains 4 sets of PWM modules. Each set of PWM module has two output ports:

- PWM 1 (GPIO 0 and 1)
- PWM 2 (GPIO 2 and 3)
- PWM 3 (GPIO 4 and 5)
- PWM 4 (GPIO 6 and 7).

The two outputs of each PWM module usually are complementary to each other. For example PWM 1 has a positive output PWM 1A and a negative output PWM 1B, except when the PWM module is set in one of a few special operation modes.

In SimCoder, these 4 PWM's can be used in the following ways:

- One 3-phase PWM generators: PWM 123 (consisting of PWM 1, 2, and 3);
- Four 2-phase PWM generators: PWM 1, 2, 3, and 4, with the two outputs of each PWM generator not in a complementary way, but in special operation mode.
- 1-phase PWM generators: PWM 1, 2, 3, and 4, with two outputs complementary to each other.
- 1-phase PWM generators with phase shift: PWM 2, 3, and 4, with two outputs complementary to each other.

These PWM generators can trigger the A/D converter, and use trip-zone signals.

Beside the PWM generators described above, there is also one APWM generator that uses the same resource as the capture. This PWM generator has restricted functionality as compared to the 4 PWM generators (PWM 1 to 4) as they can not trigger the A/D converter and can not use trip-zone signals. Also, because of the common resources, when a particular port is used for the capture, it can not be used for the PWM generator.

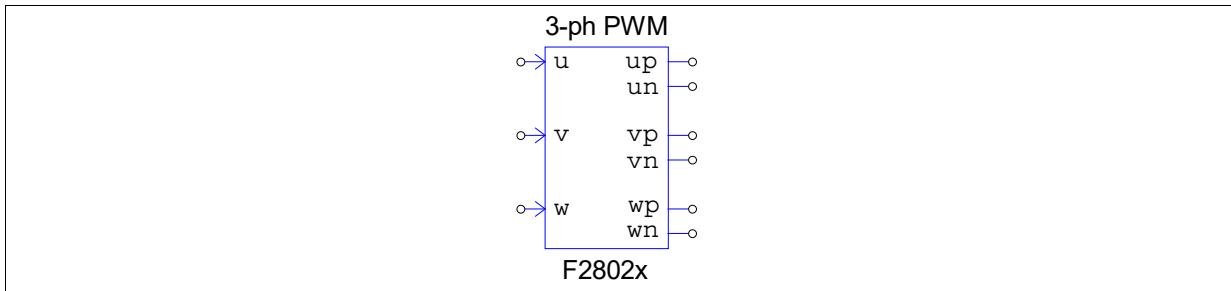
Note that all the PWM generators in SimCoder include one switching period delay internally. That is, the input value of a PWM generator is delayed by one cycle before it is used to update the PWM output. This delay is needed to simulate the delay inherent in the DSP hardware implementation.

PWM generators have a parameter called "PWM Freq. Scaling Factor". It can be set to 1 to 100. The hardware limit is 3. If the scaling factor is greater than 3, PWM will use an unused PWM to generator interrupt at the sampling frequency. This unused PWM is only used to generate a periodic interrupt, and its outputs can still be used for other functions. If there is no unused PWM in the system, a timer will be used.

9.4.1 3-Phase PWM

In the 3-phase PWM generator image, "u", "v", and "w" refer to the three phases (alternatively they are called Phase "a", "b", and "c"). The letter "p" refers to the positive output, and "n" refers to the negative output. For example, for 3-phase PWM 123, "up" is PWM1A, and "un" is PWM1B.

Image:



Attributes:

Parameters	Description
PWM Source	Source of the PWM generator. It can be "3-phase PWM 123" that uses PWM 1 to 3.
Dead Time	The dead time T_d for the PWM generator, in sec.
Sampling Frequency	Sampling frequency of the PWM generator, in Hz. The calculation is done and the PWM signal duty cycle is updated based on this frequency.
PWM Freq. Scaling Factor	The ratio between the PWM switching frequency and the sampling frequency. It can be from 1 to 100. For example, if the sampling frequency is 50 kHz and the scaling factor is 3, the PWM switching frequency will be 150 kHz. That is switches will operate at 150 kHz. But gating signals will be updated at 50 kHz, or once per 3 switching cycles.
Carrier Wave Type	Carrier wave type and the initial PWM output state. It can be one of the following: <ul style="list-style-type: none"> - <i>Triangular (start low)</i>: Triangular wave, and the initial PWM output state is low. - <i>Triangular (start high)</i>: Triangular wave, and the initial output state is high. - <i>Sawtooth (start low)</i>: Sawtooth wave, and the initial output state is low. - <i>Sawtooth (start high)</i>: Sawtooth wave, with the initial output state is high.
Trigger ADC	Setting whether for the PWM generator to trigger the A/D converter. It can be one of the following: <ul style="list-style-type: none"> - <i>Do not trigger ADC</i>: PWM does not trigger the A/D converter. - <i>Trigger ADC</i>: PWM will trigger A/D converter.
ADC Trigger Position	A/D trigger position ranges from 0 to a value less than 1. When it is 0, the A/D converter is triggered at the beginning of the PWM cycle, and when it is 0.5, the A/D converter is triggered at the 180° position of the PWM cycle.
Use Trip-Zone i	Define whether the PWM generator uses the i_{th} trip-zone signal or not, where i ranges from 1 to 6. It can be one of the following: <ul style="list-style-type: none"> - <i>Disable Trip-Zone i</i>: Disable the i_{th} trip-zone signal. - <i>One shot</i>: The PWM generator uses the trip-zone signal in the one-shot mode. Once triggered, the PWM must be started manually. - <i>Cycle by cycle</i>: The PWM generator uses the trip-zone signal in the cycle-by-cycle basis. The trip-zone signal is effective within the current cycle, and PWM will automatically re-start in the next cycle.

PWMA DC Trip Src1(DCAH)	Digital compare (DC) trip source DCAH for PWMA. For a 3-phase PWM generator, PWMA refers to the outputs "up", "vp", and "wp" for the 3 top switches. The PWM channel may have up to two DC trip sources: DCAH and DCAL. The trip source can be one of the following: <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>Trip-zone 1</i>: PWM uses trip-zone 1 as digital compare input signal. - <i>Trip-zone 2</i>: PWM uses trip-zone 2 as digital compare input signal. - <i>Trip-zone 3</i>: PWM uses trip-zone 3 as digital compare input signal. - <i>Comparator 1</i>: PWM uses Comparator 1 as digital compare input signal. - <i>Comparator 2</i>: PWM uses Comparator 2 as digital compare input signal. - <i>Comparator 3</i>: PWM uses Comparator 3 as digital compare input signal.
PWMA DC Trip Src1(DCAL)	Digital compare (DC) trip source DCAL for PWMA. The trip source can be one of the following: <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>Trip-zone 1</i>: PWM uses trip-zone 1 as digital compare input signal. - <i>Trip-zone 2</i>: PWM uses trip-zone 2 as digital compare input signal. - <i>Trip-zone 3</i>: PWM uses trip-zone 3 as digital compare input signal. - <i>Comparator 1</i>: PWM uses Comparator 1 as digital compare input signal. - <i>Comparator 2</i>: PWM uses Comparator 2 as digital compare input signal. - <i>Comparator 3</i>: PWM uses Comparator 3 as digital compare input signal.
PWMA 1-shot Evt (DCAEVT1)	Define how the one-shot signal is used for the DC trip signal of PWMA. The active level of the DC trip signal can be selected from the following: <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>Trip source 1 is low</i>: PWM is tripped if the source signal 1 is low. - <i>Trip source 1 is high</i>: PWM is tripped if the source signal 1 is high. - <i>Trip source 2 is low</i>: PWM is tripped if the source signal 2 is low. - <i>Trip source 3 is high</i>: PWM is tripped if the source signal 2 is high. - <i>Trip source 1 low & source 2 high</i>: PWM is tripped if the source 1 signal is low and at the same time source 2 signal is high.
PWMA CBC Evt (DCAEVT1)	Define how the cycle-by-cycle signal is used for the DC trip signal of PWMA. The active level of the DC trip signal can be selected from the following: <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>Trip source 1 is low</i>: PWM is tripped if the source signal 1 is low. - <i>Trip source 1 is high</i>: PWM is tripped if the source signal 1 is high. - <i>Trip source 2 is low</i>: PWM is tripped if the source signal 2 is low. - <i>Trip source 3 is high</i>: PWM is tripped if the source signal 2 is high. - <i>Trip source 1 low & source 2 high</i>: PWM is tripped if the source 1 signal is low and at the same time source 2 signal is high.
PWMB DC Trip Src1(DCBH)	Digital compare (DC) trip source DCBH for PWMB. For a 3-phase PWM generator, PWMB refers to the outputs "un", "vn", and "wn" for the 3 bottom switches. The PWM channel may have up to two DC trip sources: DCBH and DCBL. The trip source can be one of the following: <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>Trip-zone 1</i>: PWM uses trip-zone 1 as digital compare input signal. - <i>Trip-zone 2</i>: PWM uses trip-zone 2 as digital compare input signal. - <i>Trip-zone 3</i>: PWM uses trip-zone 3 as digital compare input signal. - <i>Comparator 1</i>: PWM uses Comparator 1 as digital compare input signal. - <i>Comparator 2</i>: PWM uses Comparator 2 as digital compare input signal. - <i>Comparator 3</i>: PWM uses Comparator 3 as digital compare input signal.

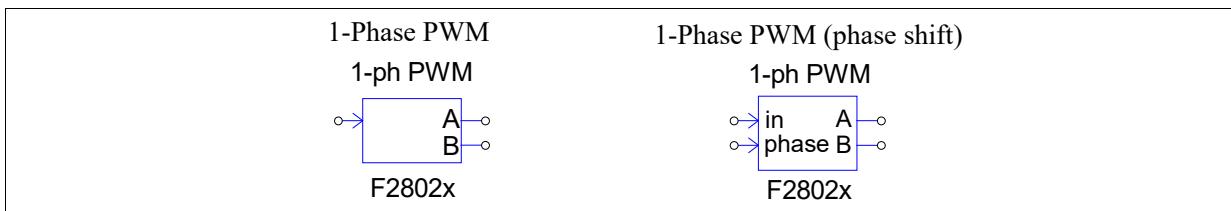
PWMB DC Trip Src1(DCBL)	Digital compare (DC) trip source DCBL for PWMB. The trip source can be one of the following: <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>Trip-zone 1</i>: PWM uses trip-zone 1 as digital compare input signal. - <i>Trip-zone 2</i>: PWM uses trip-zone 2 as digital compare input signal. - <i>Trip-zone 3</i>: PWM uses trip-zone 3 as digital compare input signal. - <i>Comparator 1</i>: PWM uses Comparator 1 as digital compare input signal. - <i>Comparator 2</i>: PWM uses Comparator 2 as digital compare input signal. - <i>Comparator 3</i>: PWM uses Comparator 3 as digital compare input signal.
PWMB 1-shot Evt (DCBEVT1)	Define how the one-shot signal is used for the DC trip signal of PWMB. The active level of the DC trip signal can be selected from the following: <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>Trip source 1 is low</i>: PWM is tripped if the source signal 1 is low. - <i>Trip source 1 is high</i>: PWM is tripped if the source signal 1 is high. - <i>Trip source 2 is low</i>: PWM is tripped if the source signal 2 is low. - <i>Trip source 3 is high</i>: PWM is tripped if the source signal 2 is high. - <i>Trip source 1 low & source 2 high</i>: PWM is tripped if the source 1 signal is low and at the same time source 2 signal is high.
PWMB CBC Evt (DCBEVT1)	Define how the cycle-by-cycle signal is used for the DC trip signal of PWMB. The active level of the DC trip signal can be selected from the following: <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>Trip source 1 is low</i>: PWM is tripped if the source signal 1 is low. - <i>Trip source 1 is high</i>: PWM is tripped if the source signal 1 is high. - <i>Trip source 2 is low</i>: PWM is tripped if the source signal 2 is low. - <i>Trip source 3 is high</i>: PWM is tripped if the source signal 2 is high. - <i>Trip source 1 low & source 2 high</i>: PWM is tripped if the source 1 signal is low and at the same time source 2 signal is high.
DC Event Filter Source	Source of the digital compare event filter. It can be one of the following: <ul style="list-style-type: none"> - <i>Do not use</i>: Do not use DC event filter. - <i>DCAEVT1</i>: Use DCAEVT1 as the filter source. - <i>DCAEVT2</i>: Use DCAEVT2 as the filter source. - <i>DCBEVT1</i>: Use DCBEVT1 as the filter source. - <i>DCBEVT2</i>: Use DCBEVT2 as the filter source. - <i>DCAEVT2</i>: PWM is tripped if the source signal 1 is high.
Blanking Window Pos (us)	Blanking window start position in a PWM period, in us.
Blanking Window Width (us)	Width of the blanking window, in us. The width is limited by the hardware, and can be calculated as below: $255 / \text{CPU Frequency}$ <p>For example, when the CPU speed is 90MHz, the width range is 0 to 2.83us.</p>
Blanking Window Range	Specify how the blanking window is applied. It can be one of the following: <ul style="list-style-type: none"> - <i>In the window</i>: The blanking action is applied with the window defined (from the start position for a window width defined) - <i>Out of window</i>: The blanking action is applied outside the window defined.
Applying Event Filtering	Specify how event filtering is applied to digital compare events. The event filtering can be applied to any combinations of the following digital compare events: DCAEVT1, DCAEVT2, DCBEVT1, and DCBEVT2

Trip Action	Define how the PWM generator responds to the trip action. It can be one of the following: <ul style="list-style-type: none"> - <i>High impedance</i>: PWM outputs in high impedance - <i>PWM A high & B low</i>: Set PWM A high and B low. - <i>PWM A low & B high</i>: Set PWM A low and B high. - <i>No action</i>: No action taken.
Peak-to-Peak Value	Peak-to-peak value V_{pp} of the carrier wave
Offset Value	DC offset value V_{offset} of the carrier wave
Initial Input Value u, v, w	Initial value of 3-phase inputs u , v , and w
Start PWM at Beginning	When it is set to <i>Start</i> , PWM will start right from the beginning. If it is set to <i>Do not start</i> , one needs to start PWM using the Start PWM block.
Simulation Output Mode	The simulation output mode can be set to <i>Switching mode</i> or <i>Average mode</i> . When it is set to "Switching mode", the outputs of the PWM block are PWM signals. When it is set to "Average mode", the outputs of the PWM block are average mode signals. In the average mode, if the carrier wave is from negative to positive, and the absolute values of the negative peak and the positive peak are equal (for example, the carrier wave is from -1 to +1, or from -5 to +5), the modulation is considered as an ac signal modulation. Otherwise, the modulation is considered as a dc signal modulation. For example, modulation in a 3-phase or single-phase inverter is an ac modulation, and modulation in a buck converter is a dc modulation. In the ac signal modulation, if the input u of the PWM block is V_u , the output up and un in average mode will be: $V_{up} = V_u / (V_{pp} + V_{offset})$ $V_{un} = -V_{up}$ In this case, V_u is between $-(V_{pp} + V_{offset})$ and $V_{pp} + V_{offset}$ and V_{up} is between -1 to +1. In the dc signal modulation, the output up and un in average mode will be: $V_{up} = (V_u - V_{offset}) / V_{pp}$ $V_{un} = 1 - V_{up}$ In this case, V_u is between V_{offset} and $V_{pp} + V_{offset}$ and V_{up} is between 0 to +1. When it is set to the average mode, the PWM block outputs can be connected to a converter/inverter in the average mode model.

9.4.2 1-Phase PWM and 1-Phase PWM (phase shift)

The attributes for the **1-Phase PWM** block and **1-phase PWM (phase shift)** block are mostly the same. The difference is that the 1-Phase PWM block defines the phase shift through a parameter, while the 1-Phase PWM (phase shift) block reads the phase shift from an external input (labeled as "phase" in the image). The phase shift is in degree.

Image:



Attributes:

Parameters	Description
PWM Source	Source of the PWM generator. Without phase shift, it can be PWM 1 to PWM 4. With phase shift, it can be PWM 2 to PWM 4.
Output Mode	Output mode of the PWM generator. It can be one of the following: <ul style="list-style-type: none"> - <i>Use PWM A&B</i>: Both PWM outputs A and B are used, and they are complementary. - <i>Use PWM A</i>: Only PWM output A is used. - <i>Use PWM B</i>: Only PWM output B is used.
Dead Time	Dead time T_d for the PWM generator, in sec.
Sampling Frequency	Sampling frequency of the PWM generator, in Hz. The calculation is done and the PWM signal duty cycle is updated based on this frequency.
PWM Freq. Scaling Factor	The ratio between the PWM switching frequency and the sampling frequency. It can be from 1 to 100. For example, if the sampling frequency is 50 kHz and the scaling factor is 3, the PWM switching frequency will be 150 kHz. That is switches will operate at 150 kHz. But gating signals will be updated at 50 kHz, or once per 3 switching cycles.
Carrier Wave Type	Carrier wave type and the initial PWM output state. It can be one of the following: <ul style="list-style-type: none"> - <i>Triangular (start low)</i>: Triangular wave, and the initial PWM output state is low. - <i>Triangular (start high)</i>: Triangular wave, and the initial output state is high. - <i>Sawtooth (start low)</i>: Sawtooth wave, and the initial output state is low. - <i>Sawtooth (start high)</i>: Sawtooth wave, and the initial output state is high.
Trigger ADC	Setting whether for the PWM generator to trigger the A/D converter. It can be one of the following: <ul style="list-style-type: none"> - <i>Do not trigger ADC</i>: PWM does not trigger the A/D converter. - <i>Trigger ADC Group A</i>: PWM will trigger Group A of the A/D converter. - <i>Trigger ADC Group B</i>: PWM will trigger Group B of the A/D converter. - <i>Trigger ADC Group A&B</i>: PWM will trigger both Group A and B of the A/D converter.
ADC Trigger Position	A/D trigger position ranges from 0 to a value less than 1. When it is 0, the A/D converter is triggered at the beginning of the PWM cycle, and when it is 0.5, the A/D converter is triggered at the 180° position of the PWM cycle.
Use Trip-Zone i	Define whether the PWM generator uses the i_{th} trip-zone signal or not, where i ranges from 1 to 6. It can be one of the following: <ul style="list-style-type: none"> - <i>Disable Trip-Zone i</i>: Disable the i_{th} trip-zone signal. - <i>One shot</i>: The PWM generator uses the trip-zone signal in the one-shot mode. Once triggered, the PWM must be started manually. - <i>Cycle by cycle</i>: The PWM generator uses the trip-zone signal in the cycle-by-cycle basis. The trip-zone signal is effective within the current cycle, and PWM will automatically re-start in the next cycle.

PWMA DC Trip Src1(DCAH)	Digital compare (DC) trip source DCAH for PWMA. The PWM channel may have up to two DC trip sources: DCAH and DCAL. The trip source can be one of the following: <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>Trip-zone 1</i>: PWM uses trip-zone 1 as digital compare input signal. - <i>Trip-zone 2</i>: PWM uses trip-zone 2 as digital compare input signal. - <i>Trip-zone 3</i>: PWM uses trip-zone 3 as digital compare input signal. - <i>Comparator 1</i>: PWM uses Comparator 1 as digital compare input signal. - <i>Comparator 2</i>: PWM uses Comparator 2 as digital compare input signal. - <i>Comparator 3</i>: PWM uses Comparator 3 as digital compare input signal.
PWMA DC Trip Src1(DCAL)	Digital compare (DC) trip source DCAL for PWMA. The trip source can be one of the following: <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>Trip-zone 1</i>: PWM uses trip-zone 1 as digital compare input signal. - <i>Trip-zone 2</i>: PWM uses trip-zone 2 as digital compare input signal. - <i>Trip-zone 3</i>: PWM uses trip-zone 3 as digital compare input signal. - <i>Comparator 1</i>: PWM uses Comparator 1 as digital compare input signal. - <i>Comparator 2</i>: PWM uses Comparator 2 as digital compare input signal. - <i>Comparator 3</i>: PWM uses Comparator 3 as digital compare input signal.
PWMA 1-shot Evt (DCAEVT1)	Define how the one-shot signal is used for the DC trip signal of PWMA. The active level of the DC trip signal can be selected from the following: <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>Trip source 1 is low</i>: PWM is tripped if the source signal 1 is low. - <i>Trip source 1 is high</i>: PWM is tripped if the source signal 1 is high. - <i>Trip source 2 is low</i>: PWM is tripped if the source signal 2 is low. - <i>Trip source 3 is high</i>: PWM is tripped if the source signal 2 is high. - <i>Trip source 1 low & source 2 high</i>: PWM is tripped if the source 1 signal is low and at the same time source 2 signal is high.
PWMA CBC Evt (DCAEVT1)	Define how the cycle-by-cycle signal is used for the DC trip signal of PWMA. The active level of the DC trip signal can be selected from the following: <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>Trip source 1 is low</i>: PWM is tripped if the source signal 1 is low. - <i>Trip source 1 is high</i>: PWM is tripped if the source signal 1 is high. - <i>Trip source 2 is low</i>: PWM is tripped if the source signal 2 is low. - <i>Trip source 3 is high</i>: PWM is tripped if the source signal 2 is high. - <i>Trip source 1 low & source 2 high</i>: PWM is tripped if the source 1 signal is low and at the same time source 2 signal is high.
PWMB DC Trip Src1(DCBH)	Digital compare (DC) trip source DCBH for PWMB. The PWM channel may have up to two DC trip sources: DCBH and DCBL. The trip source can be one of the following: <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>Trip-zone 1</i>: PWM uses trip-zone 1 as digital compare input signal. - <i>Trip-zone 2</i>: PWM uses trip-zone 2 as digital compare input signal. - <i>Trip-zone 3</i>: PWM uses trip-zone 3 as digital compare input signal. - <i>Comparator 1</i>: PWM uses Comparator 1 as digital compare input signal. - <i>Comparator 2</i>: PWM uses Comparator 2 as digital compare input signal. - <i>Comparator 3</i>: PWM uses Comparator 3 as digital compare input signal.

PWMB DC Trip Src1(DCBL)	Digital compare (DC) trip source DCBL for PWMB. The trip source can be one of the following: <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>Trip-zone 1</i>: PWM uses trip-zone 1 as digital compare input signal. - <i>Trip-zone 2</i>: PWM uses trip-zone 2 as digital compare input signal. - <i>Trip-zone 3</i>: PWM uses trip-zone 3 as digital compare input signal. - <i>Comparator 1</i>: PWM uses Comparator 1 as digital compare input signal. - <i>Comparator 2</i>: PWM uses Comparator 2 as digital compare input signal. - <i>Comparator 3</i>: PWM uses Comparator 3 as digital compare input signal.
PWMB 1-shot Evt (DCBEVT1)	Define how the one-shot signal is used for the DC trip signal of PWMB. The active level of the DC trip signal can be selected from the following: <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>Trip source 1 is low</i>: PWM is tripped if the source signal 1 is low. - <i>Trip source 1 is high</i>: PWM is tripped if the source signal 1 is high. - <i>Trip source 2 is low</i>: PWM is tripped if the source signal 2 is low. - <i>Trip source 3 is high</i>: PWM is tripped if the source signal 2 is high. - <i>Trip source 1 low & source 2 high</i>: PWM is tripped if the source 1 signal is low and at the same time source 2 signal is high.
PWMB CBC Evt (DCBEVT1)	Define how the cycle-by-cycle signal is used for the DC trip signal of PWMB. The active level of the DC trip signal can be selected from the following: <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>Trip source 1 is low</i>: PWM is tripped if the source signal 1 is low. - <i>Trip source 1 is high</i>: PWM is tripped if the source signal 1 is high. - <i>Trip source 2 is low</i>: PWM is tripped if the source signal 2 is low. - <i>Trip source 3 is high</i>: PWM is tripped if the source signal 2 is high. - <i>Trip source 1 low & source 2 high</i>: PWM is tripped if the source 1 signal is low and at the same time source 2 signal is high.
DC Event Filter Source	Source of the digital compare event filter. It can be one of the following: <ul style="list-style-type: none"> - <i>Do not use</i>: Do not use DC event filter. - <i>DCAEVT1</i>: Use DCAEVT1 as the filter source. - <i>DCAEVT2</i>: Use DCAEVT2 as the filter source. - <i>DCBEVT1</i>: Use DCBEVT1 as the filter source. - <i>DCBEVT2</i>: Use DCBEVT2 as the filter source. - <i>DCAEVT2</i>: PWM is tripped if the source signal 1 is high.
Blanking Window Pos (us)	Blanking window start position in a PWM period, in us.
Blanking Window Width (us)	Width of the blanking window, in us. The width is limited by the hardware, and can be calculated as below: $255 / \text{CPU Frequency}$ <p>For example, when the CPU speed is 90MHz, the width range is 0 to 2.83us.</p>
Blanking Window Range	Specify how the blanking window is applied. It can be one of the following: <ul style="list-style-type: none"> - <i>In the window</i>: The blanking action is applied with the window defined (from the start position for a window width defined) - <i>Out of window</i>: The blanking action is applied outside the window defined.
Applying Event Filtering	Specify how event filtering is applied to digital compare events. The event filtering can be applied to any combinations of the following digital compare events: DCAEVT1, DCAEVT2, DCBEVT1, and DCBEVT2

Trip Action	Define how the PWM generator responds to the trip action. It can be one of the following: <ul style="list-style-type: none"> - <i>High impedance</i>: PWM outputs in high impedance - <i>PWM A high & B low</i>: Set PWM A high and B low. - <i>PWM A low & B high</i>: Set PWM A low and B high. - <i>No action</i>: No action taken.
Peak-to-Peak Value	Peak-to-peak value V_{pp} of the carrier wave
Offset Value	DC offset value V_{offset} of the carrier wave
Phase Shift	Phase shift of the output with respect to the reference PWM generator output, in deg. (for 1-phase PWM generator without external phase shift input)
Initial Input Value	Initial value of the input
Use HRPWM	Define the high-resolution PWM. It can be one of the following: <ul style="list-style-type: none"> - <i>Do not use HRPWM</i>: Do not use high-resolution PWM - <i>Use HRPWM without calibration</i>: Use high-resolution PWM without calibration - <i>Use HRPWM with calibration</i>: Use high-resolution PWM with calibration
Start PWM at Beginning	When it is set to <i>Start</i> , PWM will start right from the beginning. If it is set to <i>Do not start</i> , one needs to start PWM using the Start PWM block.
Simulation Output Mode	The simulation output mode can be set to <i>Switching mode</i> or <i>Average mode</i> . When it is set to "Switching mode", the outputs of the PWM block are PWM signals. When it is set to "Average mode", the outputs of the PWM block are average mode signals. In the average mode, if the carrier wave is from negative to positive, and the absolute values of the negative peak and the positive peak are equal (for example, the carrier wave is from -1 to +1, or from -5 to +5), the modulation is considered as an ac signal modulation. Otherwise, the modulation is considered as a dc signal modulation. For example, modulation in a 3-phase or single-phase inverter is an ac modulation, and modulation in a buck converter is a dc modulation. In the ac signal modulation, if the input of the PWM block is V_{in} , the output A and B in average mode will be: $V_A = V_{in} / (V_{pp} + V_{offset})$ $V_B = -V_A$ In this case, V_{in} is between $-(V_{pp} + V_{offset})$ and $V_{pp} + V_{offset}$, and V_A and V_B are between -1 to +1. In the dc signal modulation, the output A and B in average mode will be: $V_A = (V_{in} - V_{offset}) / V_{pp}$ $V_B = 1 - V_A$ In this case, V_{in} is between V_{offset} and $V_{pp} + V_{offset}$, and V_A and V_B are between 0 to +1. When it is set to the average mode, the PWM block outputs can be connected to a converter/inverter in the average mode model.

Phase Shift

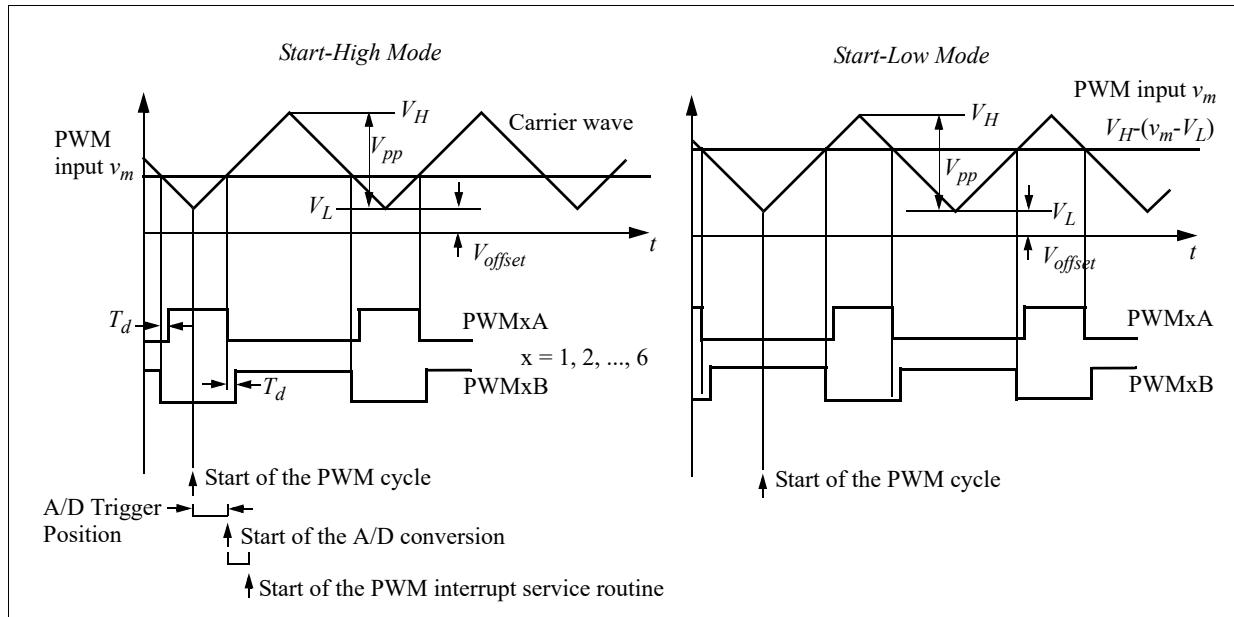
A 1-phase PWM generator can generate PWM signal that is phase shifted with respect to another PWM signal. The way how PWM blocks are defined for phase shift is explained in Section 9.4.5.

The phase shift value is in degrees. When the value is -30° , the output will be shifted to the right (lagging) by 30° of the switching cycle with respect to the reference PWM generator output. This is equivalent to shifting the

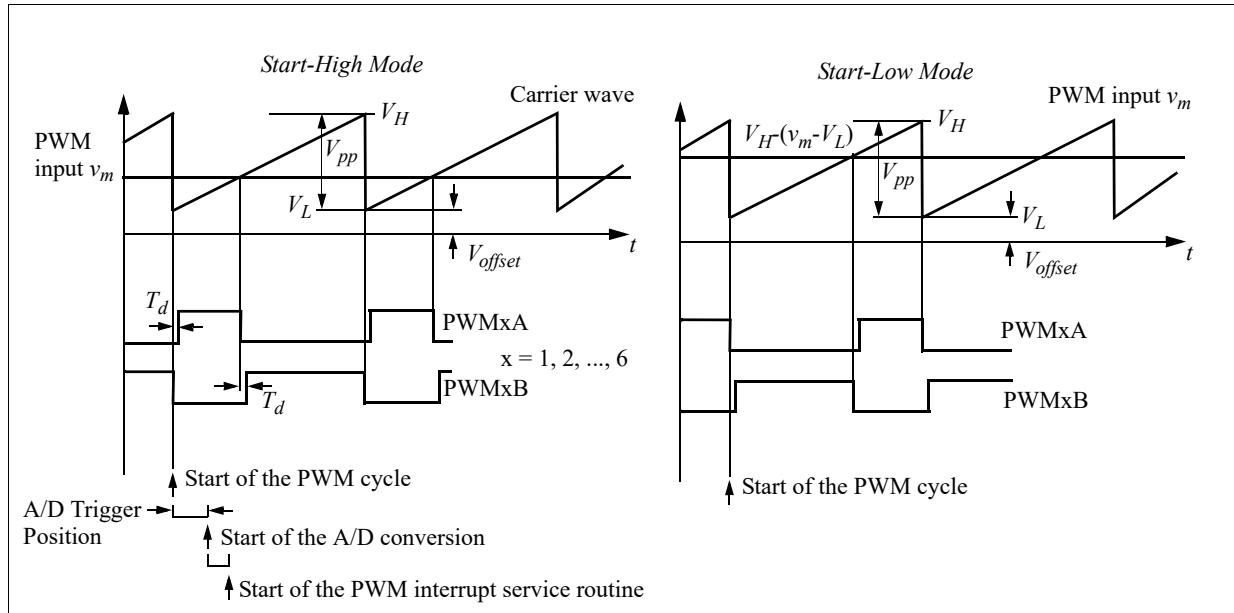
PWM carrier wave to the right by 30° . When the phase value is 30° , the output will be shifted to the left (leading) by 30° .

Carrier Wave

There are two types of carrier waveforms: triangular wave (with equal rising and falling slope intervals) and sawtooth wave. In addition, there are two operation modes: start-low and start-high modes, as explained below. The input and output waveforms of a PWM generator with the triangular carrier wave are shown below:



The input and output waveforms of a PWM generator with the sawtooth carrier wave are shown below:



The figures above show how the dead time is defined, and the time sequence when the PWM generator triggers the A/D converter. If triggering the A/D converter is selected, from the start of the PWM cycle, after a certain delay defined by the A/D trigger position, the A/D conversion will start. After the A/D conversion is completed, the PWM interrupt service routine will start.

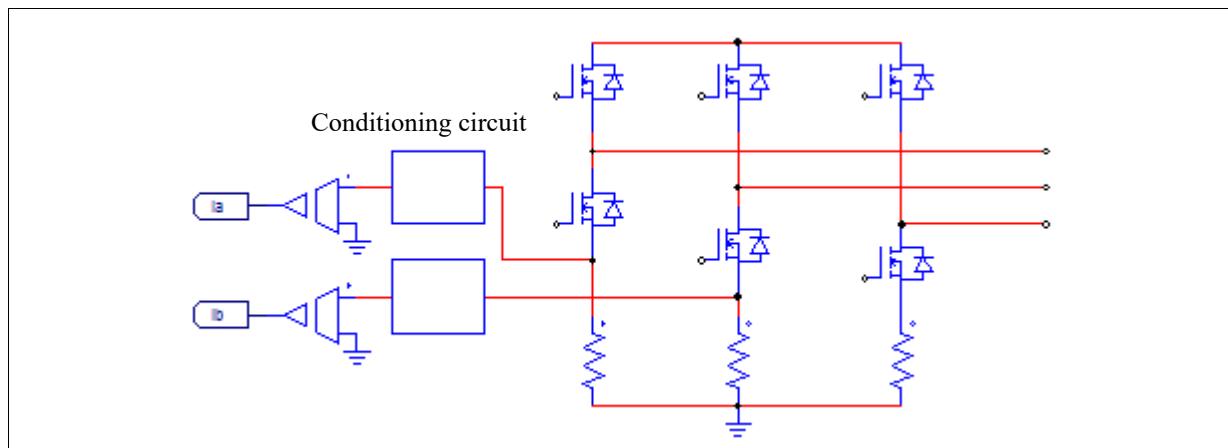
If the PWM generator does not trigger the A/D converter, the PWM interrupt service routine will start at the beginning of the PWM cycle.

The figures above also show how the start-high and start-low modes work. Assume that the PWM input is v_m , and the lowest value of the carrier wave is V_L and the highest value is V_H . In the start-high mode, the PWM positive output PWMA is high at the beginning of the switching cycle, and it remains high as long as the input v_m is greater than the carrier wave. For example, for a carrier wave from 0 to 1, $V_L=0$, and $V_H=1$. If $v_m=0.2$, the PWM output PWMA will remain high as long as the carrier is less than 0.2.

On the other hand, in the start-low mode, the PWM positive output PWMA is low at the beginning of the switching cycle, and it is high when the carrier wave is greater than the value $V_H(v_m-V_L)$. For example, for a carrier wave from 0 to 1, $V_L=0$, and $V_H=1$. If $v_m=0.2$, the PWM output PWMA will be high as long as the carrier is greater than 0.8.

The carrier start mode depends on how switch currents are measured. In a 3-phase inverter, for example, if top switch currents are measured, the start-high mode should be selected. This ensures that at the beginning of the cycle, the top switch gating signal is high and the current is conducting. On the other hand, if bottom switch currents are measured, the start-low mode should be selected. This ensures that at the beginning of the cycle, the top switch gating signal is low, and the bottom switch gating signal is high and the current is conducting.

For example, in the circuit below, the bottom switch currents of Phase A and B are measured. In this case, the carrier start-low mode should be selected.

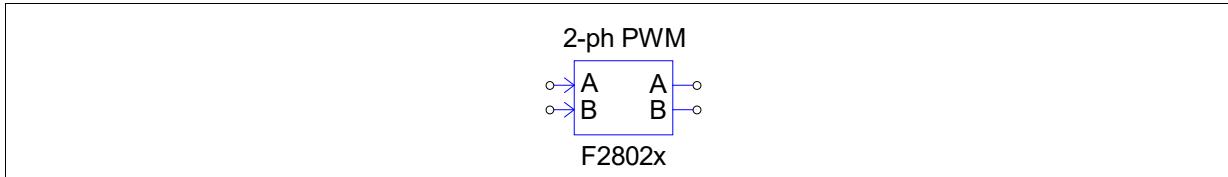


Note: In the start-low mode, the PWM input v_m is converted to $V_H(v_m-V_L)$ internally before it is compared with the carrier wave to generate the PWM signal. With the conversion, both the start-low and start-high modes will have the same duty cycle expression. For example, for a sawtooth wave with $V_L=0$ and $V_H=1$, or for a triangular wave with $V_L=-V_H$, the duty cycle D of the PWMA output in both the start-low and start-high modes is: $D = v_m/V_H$.

9.4.3 2-Phase PWM

A 2-phase PWM generator has two inputs and two outputs, and operates in one of the six pre-defined operation modes.

Image:



Attributes:

Parameters	Description
PWM Source	Source of the PWM generator. It can be PWM 1 to PWM 4.
Mode Type	Operation mode of the PWM generation. It can be one of the 6 modes. The waveforms of the 6 operation modes are described below.
Sampling Frequency	Sampling frequency of the PWM generator, in Hz. The calculation is done and the PWM signal duty cycle is updated based on this frequency.
PWM Freq. Scaling Factor	The ratio between the PWM switching frequency and the sampling frequency. It can be from 1 to 100. For example, if the sampling frequency is 50 kHz and the scaling factor is 3, the PWM switching frequency will be 150 kHz. That is switches will operate at 150 kHz. But gating signals will be updated at 50 kHz, or once per 3 switching cycles.
Trigger ADC	Setting whether for the PWM generator to trigger the A/D converter. It can be one of the following: <ul style="list-style-type: none"> - <i>Do not trigger ADC</i>: PWM does not trigger the A/D converter. - <i>Trigger ADC Group A</i>: PWM will trigger Group A of the A/D converter. - <i>Trigger ADC Group B</i>: PWM will trigger Group B of the A/D converter. - <i>Trigger ADC Group A&B</i>: PWM will trigger both Group A and B of the A/D converter.
ADC Trigger Position	A/D trigger position ranges from 0 to a value less than 1. When it is 0, the A/D converter is triggered at the beginning of the PWM cycle, and when it is 0.5, the A/D converter is triggered at the 180° position of the PWM cycle.
Use Trip-Zone <i>i</i>	Define whether the PWM generator uses the <i>i</i> th trip-zone signal or not, where <i>i</i> ranges from 1 to 6. It can be one of the following: <ul style="list-style-type: none"> - <i>Disable Trip-Zone <i>i</i></i>: Disable the <i>i</i>th trip-zone signal. - <i>One shot</i>: The PWM generator uses the trip-zone signal in the one-shot mode. Once triggered, the PWM must be started manually. - <i>Cycle by cycle</i>: The PWM generator uses the trip-zone signal in the cycle-by-cycle basis. The trip-zone signal is effective within the current cycle, and PWM will automatically re-start in the next cycle.
PWMA DC Trip Src1(DCAH)	Digital compare (DC) trip source DCAH for PWMA. The PWM channel may have up to two DC trip sources: DCAH and DCAL. The trip source can be one of the following: <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>Trip-zone 1</i>: PWM uses trip-zone 1 as digital compare input signal. - <i>Trip-zone 2</i>: PWM uses trip-zone 2 as digital compare input signal. - <i>Trip-zone 3</i>: PWM uses trip-zone 3 as digital compare input signal. - <i>Comparator 1</i>: PWM uses Comparator 1 as digital compare input signal. - <i>Comparator 2</i>: PWM uses Comparator 2 as digital compare input signal. - <i>Comparator 3</i>: PWM uses Comparator 3 as digital compare input signal.
PWMA DC Trip Src1(DCAL)	Digital compare (DC) trip source DCAL for PWMA. The trip source can be one of the following: <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>Trip-zone 1</i>: PWM uses trip-zone 1 as digital compare input signal. - <i>Trip-zone 2</i>: PWM uses trip-zone 2 as digital compare input signal. - <i>Trip-zone 3</i>: PWM uses trip-zone 3 as digital compare input signal. - <i>Comparator 1</i>: PWM uses Comparator 1 as digital compare input signal. - <i>Comparator 2</i>: PWM uses Comparator 2 as digital compare input signal. - <i>Comparator 3</i>: PWM uses Comparator 3 as digital compare input signal.

PWMA 1-shot Evt (DCAEVT1)	Define how the one-shot signal is used for the DC trip signal of PWMA. The active level of the DC trip signal can be selected from the following: <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>Trip source 1 is low</i>: PWM is tripped if the source signal 1 is low. - <i>Trip source 1 is high</i>: PWM is tripped if the source signal 1 is high. - <i>Trip source 2 is low</i>: PWM is tripped if the source signal 2 is low. - <i>Trip source 3 is high</i>: PWM is tripped if the source signal 2 is high. - <i>Trip source 1 low & source 2 high</i>: PWM is tripped if the source 1 signal is low and at the same time source 2 signal is high.
PWMA CBC Evt (DCAEVT1)	Define how the cycle-by-cycle signal is used for the DC trip signal of PWMA. The active level of the DC trip signal can be selected from the following: <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>Trip source 1 is low</i>: PWM is tripped if the source signal 1 is low. - <i>Trip source 1 is high</i>: PWM is tripped if the source signal 1 is high. - <i>Trip source 2 is low</i>: PWM is tripped if the source signal 2 is low. - <i>Trip source 3 is high</i>: PWM is tripped if the source signal 2 is high. - <i>Trip source 1 low & source 2 high</i>: PWM is tripped if the source 1 signal is low and at the same time source 2 signal is high.
PWMB DC Trip Src1(DCBH)	Digital compare (DC) trip source DCBH for PWMB. The PWM channel may have up to two DC trip sources: DCBH and DCBL. The trip source can be one of the following: <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>Trip-zone 1</i>: PWM uses trip-zone 1 as digital compare input signal. - <i>Trip-zone 2</i>: PWM uses trip-zone 2 as digital compare input signal. - <i>Trip-zone 3</i>: PWM uses trip-zone 3 as digital compare input signal. - <i>Comparator 1</i>: PWM uses Comparator 1 as digital compare input signal. - <i>Comparator 2</i>: PWM uses Comparator 2 as digital compare input signal. - <i>Comparator 3</i>: PWM uses Comparator 3 as digital compare input signal.
PWMB DC Trip Src1(DCBL)	Digital compare (DC) trip source DCBL for PWMB. The trip source can be one of the following: <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>Trip-zone 1</i>: PWM uses trip-zone 1 as digital compare input signal. - <i>Trip-zone 2</i>: PWM uses trip-zone 2 as digital compare input signal. - <i>Trip-zone 3</i>: PWM uses trip-zone 3 as digital compare input signal. - <i>Comparator 1</i>: PWM uses Comparator 1 as digital compare input signal. - <i>Comparator 2</i>: PWM uses Comparator 2 as digital compare input signal. - <i>Comparator 3</i>: PWM uses Comparator 3 as digital compare input signal.
PWMB 1-shot Evt (DCBEVT1)	Define how the one-shot signal is used for the DC trip signal of PWMB. The active level of the DC trip signal can be selected from the following: <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>Trip source 1 is low</i>: PWM is tripped if the source signal 1 is low. - <i>Trip source 1 is high</i>: PWM is tripped if the source signal 1 is high. - <i>Trip source 2 is low</i>: PWM is tripped if the source signal 2 is low. - <i>Trip source 3 is high</i>: PWM is tripped if the source signal 2 is high. - <i>Trip source 1 low & source 2 high</i>: PWM is tripped if the source 1 signal is low and at the same time source 2 signal is high.

PWMB CBC Evt (DCBEVT1)	Define how the cycle-by-cycle signal is used for the DC trip signal of PWMB. The active level of the DC trip signal can be selected from the following: <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>Trip source 1 is low</i>: PWM is tripped if the source signal 1 is low. - <i>Trip source 1 is high</i>: PWM is tripped if the source signal 1 is high. - <i>Trip source 2 is low</i>: PWM is tripped if the source signal 2 is low. - <i>Trip source 3 is high</i>: PWM is tripped if the source signal 2 is high. - <i>Trip source 1 low & source 2 high</i>: PWM is tripped if the source 1 signal is low and at the same time source 2 signal is high.
DC Event Filter Source	Source of the digital compare event filter. It can be one of the following: <ul style="list-style-type: none"> - <i>Do not use</i>: Do not use DC event filter. - <i>DCAEVT1</i>: Use DCAEVT1 as the filter source. - <i>DCAEVT2</i>: Use DCAEVT2 as the filter source. - <i>DCBEVT1</i>: Use DCBEVT1 as the filter source. - <i>DCBEVT2</i>: Use DCBEVT2 as the filter source. - <i>DCAEVT2</i>: PWM is tripped if the source signal 1 is high.
Blanking Window Pos (us)	Blanking window start position in a PWM period, in us.
Blanking Window Width (us)	Width of the blanking window, in us. The width is limited by the hardware, and can be calculated as below: $255 / \text{CPU Frequency}$ <p>For example, when the CPU speed is 90MHz, the width range is 0 to 2.83us.</p>
Blanking Window Range	Specify how the blanking window is applied. It can be one of the following: <ul style="list-style-type: none"> - <i>In the window</i>: The blanking action is applied with the window defined (from the start position for a window width defined) - <i>Out of window</i>: The blanking action is applied outside the window defined.
Applying Event Filtering	Specify how event filtering is applied to digital compare events. The event filtering can be applied to any combinations of the following digital compare events: DCAEVT1, DCAEVT2, DCBEVT1, and DCBEVT2
Trip Action	Define how the PWM generator responds to the trip action. It can be one of the following: <ul style="list-style-type: none"> - <i>High impedance</i>: PWM outputs in high impedance - <i>PWM A high & B low</i>: Set PWM A high and B low. - <i>PWM A low & B high</i>: Set PWM A low and B high. - <i>No action</i>: No action taken.
Peak Value	Peak value V_{pk} of the carrier wave
Initial Input Value A, B	Initial value of the inputs A and B.
Start PWM at Beginning	When it is set to "Start", PWM will start right from the beginning. If it is set to "Do not start", one needs to start PWM using the "Start PWM" function.

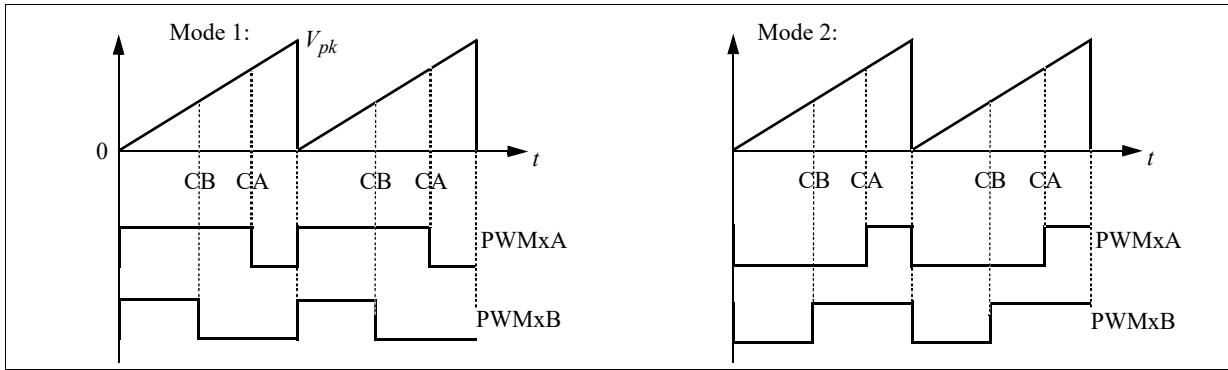
For 2-phase PWM generators, the outputs are determined based on the mode of operation, as described below. The carrier wave is either sawtooth or triangular, depending on the mode of operation. It increases from 0 to the peak value V_{pk} , and there is no dc offset.

Operation Mode 1:

The figure below on the left shows the waveforms of Mode 1. In the figure, "CA" and "CB" refer to two inputs A and B of the 2-phase PWM generator. Each input controls the turn-off time of each output.

Operation Mode 2:

The figure below on the right shows the waveforms of Mode 2. Unlike in Mode 1, each input controls the turn-on time of each output.

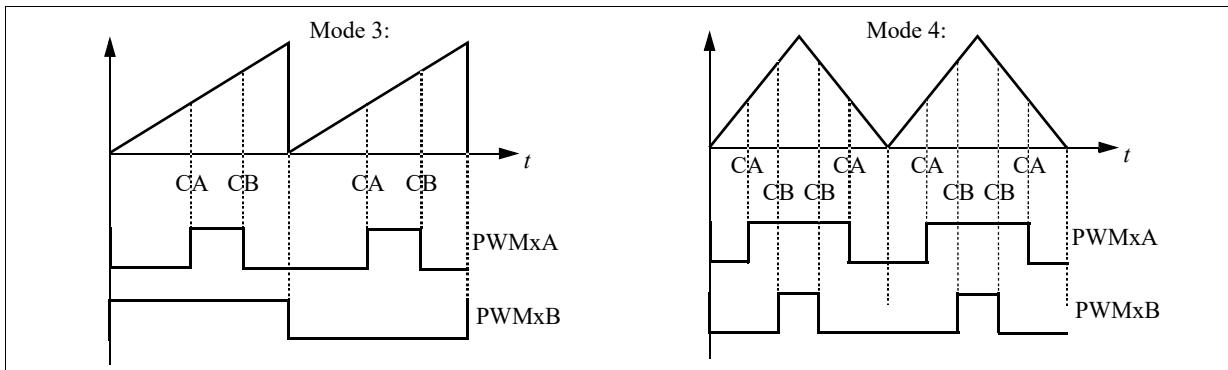


Operation Mode 3:

The figure below on the left shows the waveforms of Mode 3. Input A controls the turn-on and Input B controls the turn-off of the PWM output A. The PWM output B is on for one complete PWM cycle, and is off for the next cycle.

Operation Mode 4:

The figure below on the right shows the waveforms of Mode 4. The carrier wave is triangular. Each input controls both the turn-on and turn-off of its output.

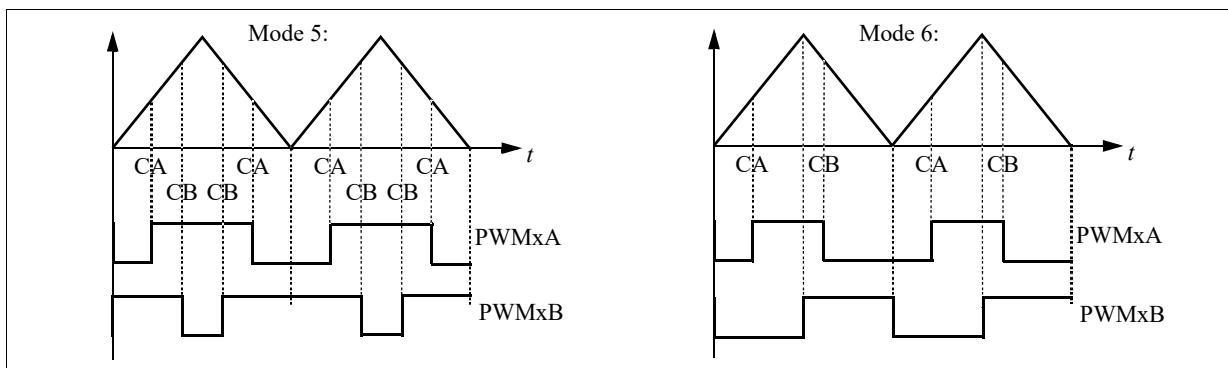


Operation Mode 5:

The figure below on the left shows the waveforms of Mode 5. The carrier wave is triangular. Similar to Mode 4, each input controls both the turn-on and turn-off of its output. Note that PWM output B is inverted in this case.

Operation Mode 6:

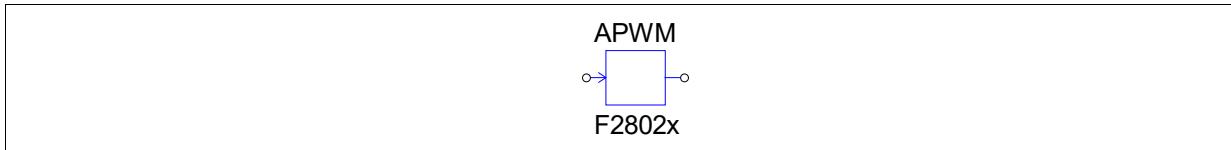
The figure below on the right shows the waveforms of Mode 6. In this mode, Input A controls the turn-on and Input B controls the turn-off of PWM output A. The PWM output B is on for the first half PWM cycle, and is off for the second half cycle.



9.4.4 Single PWM (shared with capture)

A single PWM generator, also called APWM, shares the same resource as captures. It has restricted functionality as compared to the 4 PWM generators (PWM 1 to 4) as they cannot trigger the A/D converter and cannot use trip-zone signals. Also, because of the common resource, when a particular port is used for the capture, it can not be used for the PWM generator.

Image:



Attributes:

Parameters	Description
PWM Source	Source of the PWM generator. It can be from one of the following: - APWM 1 (GPIO 5) - APWM 1 (GPIO 19)
PWM Frequency	Frequency of the PWM generator, in Hz
Carrier Wave Type	The carrier wave type and the initial PWM output state. It can be one of the following: - <i>Sawtooth (start low)</i> : Sawtooth wave, with the PWM output in the low state initially. - <i>Sawtooth (start high)</i> : Sawtooth wave, with the PWM output in the high state initially.
Stop Action	The output status when the PWM generator is stopped. It can be one of the following: - <i>Output low</i> : The PWM output will be set to low. - <i>Output high</i> : The PWM output will be set to high.
Peak-to-Peak Value	Peak-to-peak value of the carrier wave
Offset Value	DC offset value of the carrier wave
Phase Shift	Phase shift of the output with respect to the reference PWM generator, in deg.
Initial Input Value	Initial value of the input
Start PWM at Beginning	When it is set to <i>Start</i> , PWM will start right from the beginning. If it is set to <i>Do not start</i> , one needs to start PWM using the Start PWM block.

Similar to 1-phase PWM generators, an APWM generator can generate a PWM signal that has a phase shift with respect to another PWM generator. The way how PWM blocks are defined for phase shift is explained in Section 9.4.5.

9.4.5 Synchronization Between PWM Blocks

Three types of PWM blocks can be synchronized, and phase shifts can be defined between each other: **1-phase PWM**, **1-phase PWM (phase shift)**, and **APWM (or Single PWM (shared with capture))**.

A 1-phase PWM block can generate PWM signal that is phase shifted with respect to another PWM signal. There is one series in regular PWM blocks: PWM 1, 2, 3, and 4.

The definitions of the PWM blocks for phase shift are described below.

- The reference PWM and the PWM being phase shifted must be from the same series. That is, PWM 1 can be the reference, and PWM 2, 3, and 4 can be phase shifted with respect to PWM 1. Or PWM 2 can be the reference, and PWM 3 and 4 can be phase shifted with respect to PWM 2.
- The reference PWM and the PWM being shifted must be consecutive in the series, unless the skipped

PWM is not used. For example, if PWM 1, 2, and 3 are all used, but PWM 2 is not synchronized with PWM 1 and 3, this is not allowed. However, if PWM 2 is not used in the circuit, it is ok to use PWM 1 as the reference and phase shift PWM 3.

For example, the following definitions are correct, with the first PWM as the reference and the subsequent PWM blocks phase shifted:

PWM 1 (reference), PWM 2, PWM 3, PWM 4

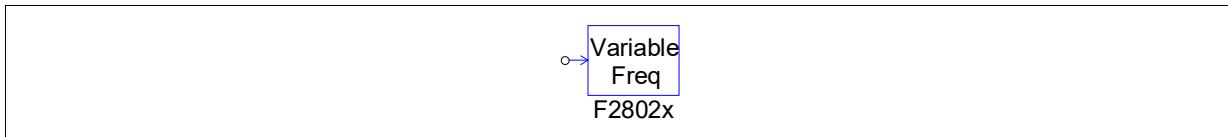
PWM 2 (reference), PWM 3

PWM 3 (reference), PWM 4

9.5 Variable Frequency PWM

The Variable Frequency PWM block provides the function to change the sampling frequency of a PWM generator. The image and parameters are shown below.

Image:



Attributes:

Parameters	Description
PWM Source	Source of the PWM generator. It can be one of the following: PWM 1, PWM 2, PWM 3, PWM 4, and 3-phase PWM 123
Adjust Interrupt Pos.	Specify if the interrupt position is adjusted with the frequency. It can be one of the following: - <i>Do not adjust</i> : The interrupt position will remain unchanged as calculated with the base frequency. - <i>Adjust</i> : The interrupt position will be recalculated at the beginning of each cycle based on the new frequency.
Adjust Ramp Compensation	Specify if the ramp compensation of the comparator DAC block is adjusted with the frequency. It can be one of the following: - <i>Do not adjust</i> : The ramp compensation will remain unchanged as calculated with the base frequency. - <i>Adjust</i> : The ramp compensation will be recalculated at the beginning of each cycle based on the new frequency.

The sampling frequency of the corresponding PWM block will be changed at the beginning of the next PWM period as follows:

$$\text{PWM_Frequency} = \text{PWM_Base Frequency} / \text{Input_Value}$$

where *PWM_Base_Frequency* is the sampling frequency of the corresponding PWM block, and *Input_Value* is the input value of this block.

If the interrupt position is to be adjusted, the interrupt position will be recalculated in each cycle. Since adjusting the interrupt position takes time, if the frequency change is small, it is recommended not to adjust the interrupt position.

Similarly, if the ramp compensation is to be adjusted, the ramp compensation will be recalculated in each cycle. Since adjusting the ramp compensation takes time, if the frequency change is small, it is recommended not to adjust the ramp compensation.

9.6 Start PWM and Stop PWM

The Start PWM and Stop PWM blocks provide the function to start/stop a PWM generator. The images and parameters are shown below.

Image:



Attributes:

Parameters	Description
PWM Source	The source of the PWM generator. It can be: PWM 1-7, 3-phase PWM 123 and PWM 456, and Capture 1.

9.7 Trip-Zone and Trip-Zone State

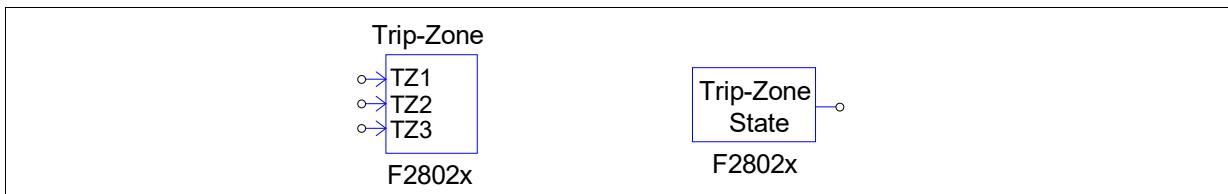
F2802x contains 6 trip-zone input signals: 3 from GPIO ports and 3 from comparators. Comparators can only be used in Digital Compare to trip PWM.

Trip-zone is used to handle external fault or trip conditions. The corresponding PWM outputs can be programmed to respond accordingly.

One trip-zone signal can be used by multiple PWM generators, and a PWM generator can use any or all of the 6 trip-zone signals. The interrupt generated by trip-zone signals are handled by the interrupt block.

The trip-zone signal triggers a trip action when the input signal is low (0). The trip-zone signals through Digital Compare trigger a trip action in specified level (either high or low)

Image:



Attributes for Trip-Zone:

Parameters	Description
Use Trip-Zone i	Specify if the i_{th} trip-zone is used.
GPIO Port for Trip-Zone i	Specify a designated GPIO port as the i_{th} trip-zone input signal. It can be one of the following: <ul style="list-style-type: none">- For trip-zone 1: GPIO12- For trip-zone 2: GPIO16 or 28- For trip-zone 3: GPIO17 or 34
Use Comparator i	Specify if the i_{th} comparator is used as the trip-zone input signal

Attributes for Trip-Zone State:

Parameters	Description
PWM Source	Source of the PWM generator. It can be: PWM 1-4 and 3-phase PWM 123.

The trip-zone interrupt can be generated in either one-shot mode or cycle-by-cycle mode, as defined in the

PWM generator parameter input. In the cycle-by-cycle mode, the interrupt only affects the PWM output within the current PWM cycle. On the other hand, in the one-shot mode, interrupt triggers a trip action when the input signal is low (0). will set the PWM output permanently, and the PWM generator must be restarted to resume the operation.

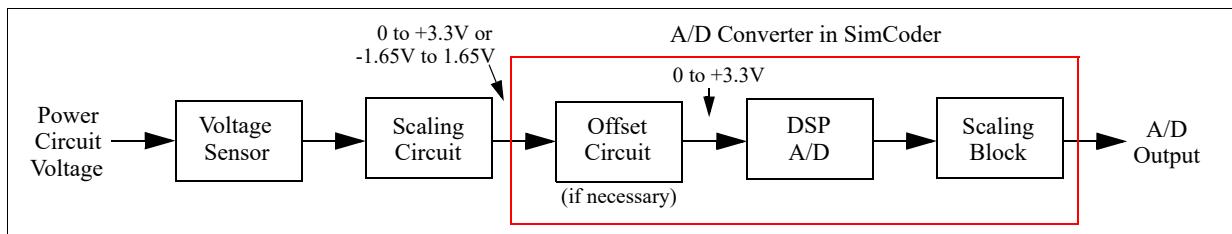
The Trip-Zone State element indicates whether the trip-zone signal is in one-shot mode or cycle-by-cycle mode when it triggers a PWM generator to generate an interrupt. When the output is 1, it means that the trip-zone signal is in one-shot mode. When the output is 0, the trip-zone signal is in cycle-by-cycle mode.

Note that when defining the interrupt block associate with trip-zone, the "Device Name" parameter of the interrupt block should be the name of the PWM generator, not the trip-zone block name. For example, if a PWM generator called "PWM_G1" uses trip-zone 1 in the trip-zone block "TZ1". The "Device Name" of the corresponding interrupt block should be "PWM_G1", not "TZ1". The "Channel Number" parameter in the interrupt block is not used in this case.

9.8 A/D Converter

F2802x provides a 13-channel 12-bit A/D converter. Note that Channels A5, B0, and B5 do not exist, and cannot be used.

Normally a power circuit quantity (voltage, current, speed, etc.) is brought to the DSP in several stages. For example, a power circuit voltage, which could be at a high level, is first converted to a control signal using a voltage sensor. A scaling circuit is then used to scale the signal, and an offset circuit is used to provide dc offset to the signal if necessary, so that the signal at the DSP A/D input is within the range of 0V and +3.3V. This signal is converted to a digital value in DSP, and a scaling block may be used to scale the value back to its original value. The complete process is shown in the diagram below.

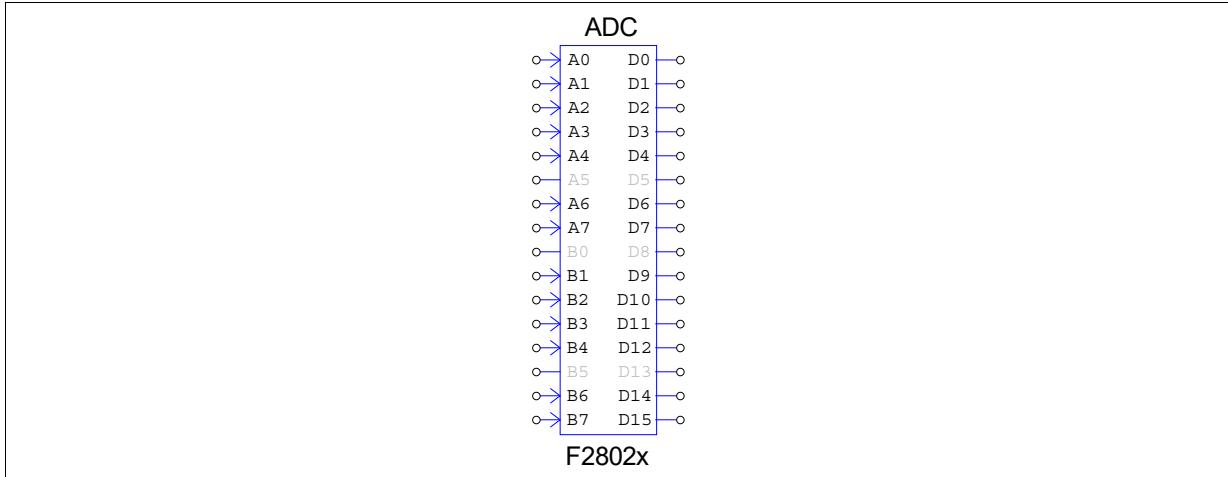


As shown above, the A/D converter element in SimCoder is not exactly the same as the physical A/D converter on the DSP. Rather, it combines the functions of an offset circuit, the DSP A/D converter, and a scaling block. This is designed for the convenience of AC system applications. It will be further explained in Section 6.5.3.

In many applications, the circuit variables to be monitored are AC signals, especially in AC motor drive systems. For each of this kind of AC signals, an offset circuit must be built in the hardware on circuit board at the input of the DSP analog input, in order to shift the signal level to the acceptable range of 0 to +3.3V. SimCoder's A/D converter provides the convenience for such cases. Instead of level-shifting and scaling the A/D output signals, user may chose to use the offset option and scaling factor in the SimCoder A/D converter, and the target code will be generated accordingly.

The image and the parameters of the A/D converter in the SimCoder library are described below. In the following description, "A/D converter" refers to the A/D converter in the SimCoder library, not the DSP A/D converter, unless otherwise stated.

Image:



Attributes:

Parameters	Description
Ch A_i or B_i Mode	Input mode of the i_{th} A/D converter channel A_i or B_i . The input mode can be one of the following: - <i>AC</i> : This option is for simulation only, not for code generation. The input range is considered from -1.65V to +1.65V. This option includes the offset circuit into the A/D converter. It provides the convenience in cases where an external level shifter is needed to shift the AC signal to the 0 to +3.3V range. - <i>DC</i> : The input is a dc value, and the range is from 0 to +3.3V.
Ch A_i or B_i Gain	Gain k of the i_{th} A/D converter channel A_i or B_i .
Conversion Order	Order of the A/D conversion. If the field is left blank (undefined), the conversion will be done based on the serial numbers of the A/D channel. For example, if A0, A2, A4, B1, and B3 are used, the conversion will be done in this order: A0, A2, A4, B1, and B3. If you wish certain channels to be performed first, you can define the order here. For example, if the conversion order is defined as: A4,A0,A2,B3,B2 The conversion will be done in the order defined, that is, A4 before A0, and A0 before A2, and so on.
ADCINT1 PIE Selection	Specify if interrupt ADCINT1 uses <i>PIE Group1</i> or <i>PIE Group10</i> .
ADCINT2 PIE Selection	Specify if interrupt ADCINT2 uses <i>PIE Group1</i> or <i>PIE Group10</i> .

An A/D converter has up to 16 channels. SimCoder divides them into groups according their sampling rates. The group with the highest sampling rate uses interrupt ADCINT1, and the group with the second highest sampling rate uses interrupt ADCINT2, etc. The two ADC groups with the highest sampling rates can choose interrupt from PIE (peripheral interrupt expansion) groups of the PIE vector table for different interrupt priority. PIE Group1 has a higher interrupt priority than PIE Group10. For example, PWM's interrupt is in PIE Group3, Its interrupt priority is lower than PIE Group1 but higher than Group10. If one wants PWM interrupt to have a higher priority than ADC interrupt, one needs to set the interrupt corresponding to the ADC channels to use PIE Group10.

Trigger Source:

The A/D converter can be triggered from multiple sources. Multiple A/D channels may share the same trigger source. Each A/D channel can be triggered by:

- One of the PWM generators,

- Timer1 or Timer2.
- More than one trigger source.

In a schematic, if a A/D channel is not associated with a PWM generator, one should insert a ZOH block at the output of the A/D channel so that SimCoder will select the timer as the trigger source.

It is not permitted to have a A/D converter channel triggered by one source, but its output signal is used in a circuit section that has a different sampling rate.

Output Scaling:

The output is scaled based on the following:

$$V_o = k * V_i$$

where V_i is the value at the input of the A/D converter.

Input Offset and Scaling:

The input of the A/D converter must stay within the input range. When the input is out of the range, it will be clamped to the limit, and a warning message will be given.

Also, the signal at the input port of the A/D converter must be scaled such that, when the input mode is DC, the maximum input voltage be scaled to +3.3V; and when the input mode is AC, the peak voltage be scaled to +/- 1.65V.

To illustrate how to use the A/D converter, two examples are given below: One with a dc input and the other with an ac input.

Assume that a power circuit voltage is a dc quantity, and the range is as follows:

$$V_{i_min} = 0 \text{ V}$$

$$V_{i_max} = 150 \text{ V}$$

The input mode of the A/D converter will be set to dc, and the input range is from 0 to +3.3V. Assume that the actual value of the voltage at a certain point is:

$$V_i = 100 \text{ V}$$

Let the voltage sensor gain be 0.01. After the voltage sensor, the maximum value and the actual value of the input become:

$$V_{i_max_s} = 150 * 0.01 = 1.5 \text{ V}$$

$$V_{i_s} = 100 * 0.01 = 1 \text{ V}$$

To utilize the full range of the DSP, a conditioning circuit with a gain of 2.2 will be used. The combined gain of the voltage sensor and the conditioning circuit becomes: $0.01 * 2.2 = 0.022$. After the conditioning circuit and at the input of the DSP A/D converter, the maximum value and the actual value of the input become:

$$V_{i_max_s_c} = 1.5 * 2.2 = 3.3 \text{ V}$$

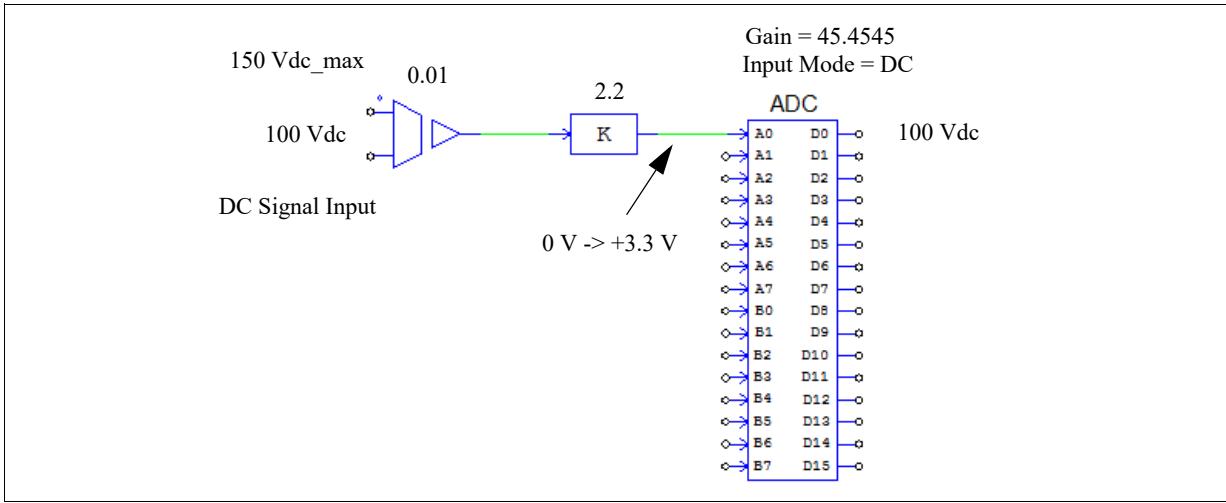
$$V_{i_s_c} = 1 * 2.2 = 2.2 \text{ V}$$

The scaling block after the DSP A/D can be selected such that the original power circuit quantity is restored. In this example, a gain of 45.4545 will be used. Note that this is the reciprocal of the combined gain of the voltage sensor and the conditioning circuit. At the A/D output, the maximum value and the actual value are:

$$V_{o_max} = 45.4545 * 3.3 = 150 \text{ V}$$

$$V_o = 45.4545 * 2.2 = 100 \text{ V}$$

The gain of the A/D channel will be set to 45.4545. The circuit connection and the settings are shown in the figure below.



Please note that, in this example, if the gain of the proportional block is changed from 2.2 to 1.1, and the A/D gain is changed from 45.4545 to 90.909, the simulation results will be the same. But the generated hardware code will not be correct. This is because the hardware code assumes that the maximum input value is scaled to +3.3V, but in this case it is only +1.5V. Therefore, one must set up the circuit such that, in the dc mode, the maximum input value is scaled to be +3.3V.

In another example, assume that a power circuit voltage is an ac quantity, and the range is as follows:

$$V_{i_max} = +/- 75 \text{ V}$$

The input mode of the A/D converter will be set to ac, and the input range is from -1.65V to +1.65V. Assume that the actual value of the voltage has a peak value of:

$$V_i = +/- 50 \text{ V}$$

Let the voltage sensor gain be 0.01. After the voltage sensor, the maximum value and the actual value of the input become:

$$V_{i_max_s} = +/- 0.75 \text{ V}$$

$$V_{i_s} = +/- 0.5 \text{ V}$$

Since the A/D converter input range is from -1.65V to +1.65V, this signal must be scaled before it is sent to the DSP. A conditioning circuit with a gain of 2.2 is needed (i.e. $1.65/0.75 = 2.2$). After the conditioning circuit and at the input of the DSP A/D converter, the maximum value and the actual value of the input become:

$$V_{i_max_s_c} = +/- 1.65 \text{ V}$$

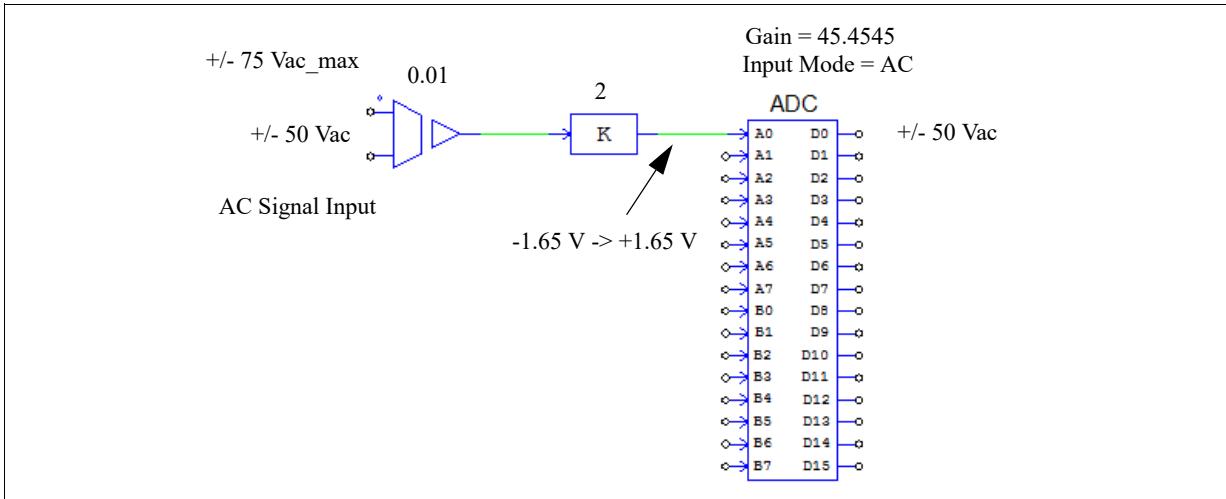
$$V_{i_s_c} = +/- 1.1 \text{ V}$$

The scaling block after the DSP A/D can be selected such that the original power circuit quantity is restored. In this example, a gain of 45.4545 will be used. Note that this is the reciprocal of the combined gain of the voltage sensor and the conditioning circuit. At the A/D output, the maximum value and the actual value are:

$$V_{o_max} = +/- 75 \text{ V}$$

$$V_o = +/- 50 \text{ V}$$

The gain of the A/D channel in PSIM will be set to 45.4545. The circuit connection and the settings are shown in the figure below.



Notice that in this circuit, the ac signal is sent to the A/D converter directly. This is because that, when the A/D input mode is set to AC, the input range is from -1.65V and +1.65V, and the function of the conditioning circuit that performs the dc offset is already included in the A/D converter block. In the actual hardware circuit, the ac signal must be scaled and offset so that the range is within 0V to +3.3V required by the A/D converter.

9.9 Comparator

F2802x supports three comparator modules. Each comparator block can have two external analog inputs, or one external analog input and one internal DAC reference for the other input. The comparator output can be sent to the PWM trip-zone and to the GPIO output.

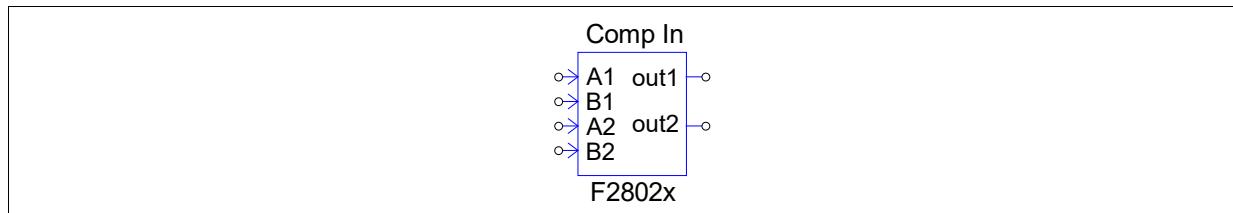
9.9.1 Comparator Input

In F2802x, there are 2 comparators. The two pairs of inputs share the same ADC/AIO ports with ADC input channels as shown below:

- Comparator 1 input A - Port ADCA2/AIO2
- Comparator 1 input B - Port ADCB2/AIO10
- Comparator 2 input A - Port ADCA4/AIO4
- Comparator 2 input B - Port ADCB4/AIO12

Only one function can be designated for each port. Simcoder will report error if a port is defined as comparator input but is used as a A/D converter channel or AIO in the same PSIM circuit schematic.

Image:



Attributes

Parameters	Description
Comparator i	<p>Define how the output of the i_{th} comparator is used. It can be one of the following:</p> <ul style="list-style-type: none"> - <i>Do not use</i>: Not used - <i>As a normal comparator</i>: Used as a normal comparator - <i>As a Trip-Zone signal</i>: Used as a trip-zone signal
Output Logic	<p>Define the comparator output logic. It can be:</p> <ul style="list-style-type: none"> - <i>High when $A > B$</i>: The output is high when Input A is greater than B. - <i>High when $A < B$</i>: The output is high when Input A is less than B.
Compare Method	<p>Define how Input A of the comparator is compared to Input B. it can be one of the following:</p> <ul style="list-style-type: none"> - <i>Compare to Input B</i>: Input A is compared to Input B where Input B is an external analog signal. - <i>Compare to Constant Value</i>: Input A is compared to a constant value. - <i>Compare to DAC output</i>: Input A is compared to a DAC output which is an internal signal.
Constant Value	<p>The constant value when <i>Compare Method</i> is defined as <i>Compare to Constant Value</i>. The range of the constant value is from 0 to 3.3V.</p>

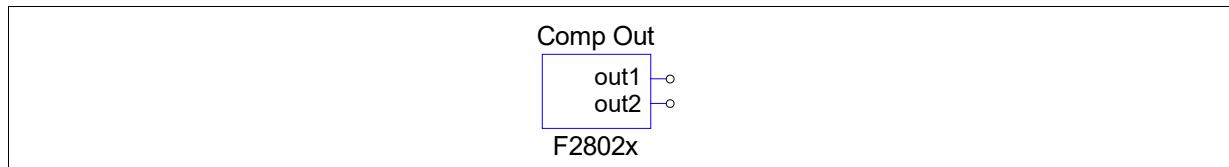
For all comparators, Input A is always from an external analog input. If Input B is also from another external analog input (if the parameter *Compare Method* is defined as *Compare to Input B*), the corresponding port must be defined as a comparator input in the Hardware Configuration block.

If the compare method is to compare to a constant value, Input B needs to be connected to ground in the schematic. If the compare method is to compare to a DAC output, Input B needs to be connected a comparator DAC output. In both cases, the corresponding ADC/AIO port can be used for other functions.

9.9.2 Comparator Output

The output of the comparator can be used as PWM trip-zone signal, as well as the GPIO output.

Image:



Attributes:

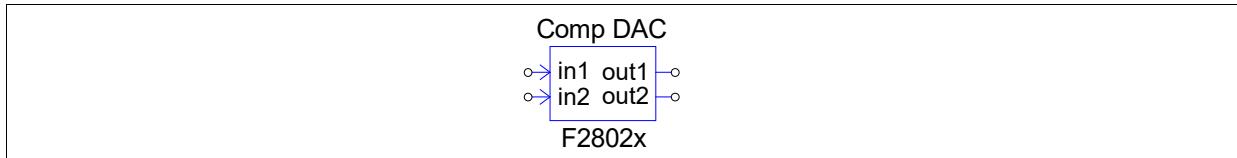
Parameters	Description
Comparator i Output	<p>Output port position of the i_{th} comparator. It can be:</p> <ul style="list-style-type: none"> - For Comparator A: GPIO1, 20, or 42 - For Comparator B: GPIO3, 21, 34, or 43 - For Comparator C: GPIO34

A comparator output block must be used together with a comparator input block. When a comparator output channel is used, the corresponding comparator input channel must be defined.

9.9.3 Comparator DAC

Each comparator block contains a 10-bit DAC reference that can be used by the inverting input (Input B) of the comparator.

Image:



Attributes:

Parameters	Description
DAC i Range	The upper limit of the input signal range of the i_{th} comparator DAC. The lower limit is 0.
Use Ramp Generator	Define if the ramp generator is used in the comparator DAC.
Total Ramp Compensation	The total compensation of the ramp generator in one PWM period. It represents the total decrease of the ramp in one cycle.

Outputs of the comparator DAC can only be connected to the corresponding inverting inputs (Input B) of the comparator in a comparator input block. That is, node out1 can only be connected to node B1 of the comparator input block, and node out2 to node B2, and node out3 to B3. Also, a comparator DAC block cannot be used alone. It must be used in conjunction with a comparator input block.

If the ramp generator is not used, the input value is applied directly to DAC output immediately. The output range is from 0 to 3.3V, and the output can be calculated as follows:

$$\text{DAC Output} = \text{DAC Input} * 3.3 / \text{DAC Range}$$

If the ramp generator is used, the input value is saved and used as the initial output value in the next PWM period. The comparator DAC output decreases linearly within the PWM period, and the total decrease is equal to the total ramp compensation value.

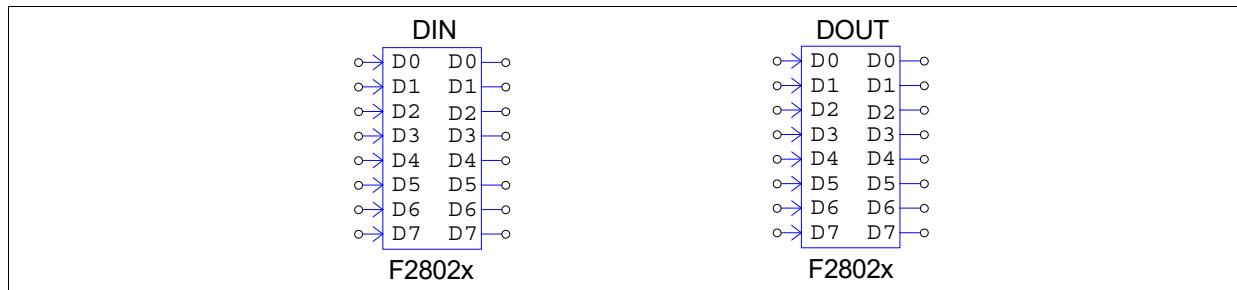
When the ramp generator is used, the sampling rate associated with the comparator DAC input must be the same as the frequency of the PWM generator that uses the comparator.

9.10 Digital Input and Digital Output

F2802x support 21 general-purpose-input-output (GPIO) ports that can be configured as either digital inputs or digital outputs. In addition, there are 6 digital analog-input-output ports (AIO 2, 4, 6, 10, 12, and 14) which also can be used as digital inputs and outputs.

In SimCoder, the digital inputs and outputs are grouped in 8-channel blocks. Multiple 8-channel digital input/output blocks can be used in the same schematic.

Image:



Attributes for Digital Input:

Parameters	Description
Port Position for Input i	The port position of the Input i . It can be one of the 21 GPIO ports or one of the 6 AIO ports.
Use as External Interrupt	Indicate if this port is used as an external interrupt input.

Attributes for Digital Output:

Parameters	Description
Port Position for Output i	The port position of the Output i . It can be one of the 21 GPIO ports or one of the 6 AIO ports.

Note that each GPIO and AIO port can be used for one function only. If an IO port is used as a digital input port, it can not be used as a digital output or any other peripheral port. For example, if Port GPIO1 is assigned as digital input and it is also used as PWM1 output, an error will be reported.

F2802x supports 3 masked external interrupts (XINT1 to XINT3). There are no dedicated pins for the external interrupts. XINT1, XINT2, and XINT3 interrupts can accept inputs from GPIO0 to GPIO31 pins.

9.11 Capture and Capture State

F2802x contains an enhanced capture module. A capture can generate interrupt, and the interrupt trigger mode is defined by the interrupt block.

Image:



Attributes for Capture:

Parameters	Description
Capture Source	Source of the capture may come from one of the GPIO ports as listed below: - Capture 1 (GPIO5) - Capture 1 (GPIO19)
Event Filter Prescale	Event filter prescale. The input signal is divided by the selected prescale.
Timer Mode	Capture counter timer mode. It can be either <i>Absolute time</i> or <i>Time difference</i> .

Attributes for Capture State:

Parameters	Description
Capture Source	Source of the capture. It has only one source <i>Capture1</i> .

The Capture State block output is either 1 or 0, where 1 means the rising edge and 0 means the falling edge.

9.12 Serial Communication Interface (SCI)

F2802x provides the function for serial communication interface (SCI). Through SCI, data inside the DSP can be transferred to a computer using an external RS-232 cable. PSIM provides all the necessary functions to transmit and receive data on both the DSP and computer sides, and to display the data on the computer. This provides a very convenient way to monitor, debug, and adjust the DSP code in real time.

For more detailed descriptions on SCI and the monitoring function, please refer to the document "Tutorial -

Using SCI for Real-Time Monitoring.pdf".

Three SCI function blocks are provided in SimCoder: *SCI Configuration*, *SCI Input*, and *SCI Output*, as described below.

9.12.1 SCI Configuration

The SCI Configuration block defines the SCI port, the communication speed, the parity check type, and the data buffer size.

Image:



Attributes:

Parameters	Description
SCI Port	Define the SCI port. These GPIO ports can be used for SCI, as listed below: SCIA: GPIO28, 7, and 18 in combination with GPIO29, 12, and 19
Speed (bps)	SCI communication speed, in bps (bits per second). A list of preset speeds is provided at 200000, 115200, 57600, 38400, 19200, or 9600 bps. Or one can specify any other speed manually.
Parity Check	The parity check setting for error check in communication. It can be either <i>None</i> , <i>Odd</i> , or <i>Even</i> .
Output Buffer Size	Size of the data buffer allocated in DSP for SCI. The buffer is located in the RAM area, and each buffer cell stores one data point which consists of three 16-bit words (that is, 6 bytes, or 48 bits, per data point).

Note that the buffer size should be properly selected. On one hand, a large buffer is preferred in order to collect more data points so that more variables can be monitored over a longer period of time. On the other hand, the internal DSP memory is limited, and the buffer should not be too large to interfere with the normal DSP operation.

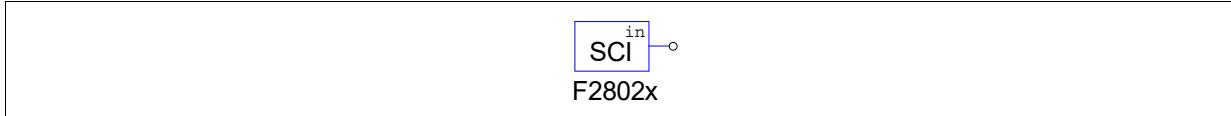
For more information on how to select the buffer size, please refer to the document "*Tutorial - Using SCI for Real-Time Monitoring.pdf*".

9.12.2 SCI Input

The SCI Input block is used to define a variable in the DSP code that can be changed. The name of the SCI input variable will appear in the DSP Oscilloscope (under the **Utilities** menu), and the value can be changed at runtime via SCI.

The SCI input block provides a convenient way to change reference, or fine tune controller parameters, for example.

Image:



Attributes:

Parameters	Description
Initial Value	Initial value of the SCI input variable.

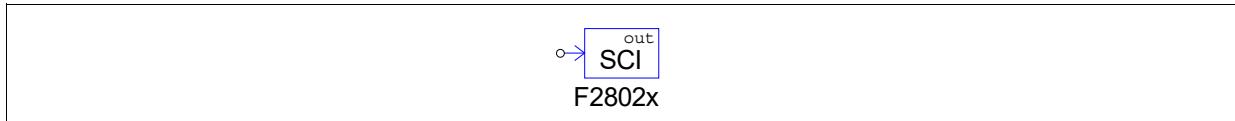
In the schematic, the SCI input behaves as a constant. While its value can be changed at runtime when the code is running on the DSP, the value will be fixed at the initial value in the simulation.

9.12.3 SCI Output

The SCI Output block is used to define a variable for display. When a SCI output block is connected to a node, the name of the SCI output block will appear in the DSP Oscilloscope (under the **Utilities** menu), and data of this variable can be transmitted from DSP to the computer via SCI at runtime, and the waveform can be displayed in the DSP Oscilloscope.

The SCI output block provides a convenient way to monitor DSP waveforms.

Image:



Attributes:

Parameters	Description
Data Point Step	It defines how frequent data is collected. If the Data Point Step is 1, every data point is collected and transmitted. If the Data Point Step is 10, for example, only one point of out every 10 points is collected and transmitted.

Note that if the Data Point Step is too small, there may be too many data points and it may not be possible to transmit them all. In this case, some data points will be discarded during the data transmission.

Also, the Data Point Step parameter is used only when the DSP Oscilloscope is in the *continuous* mode. When it is in the *snap-shot* node, this parameter is ignored and every point is collected and transmitted.

In simulation, the SCI output behaves as a voltage probe.

9.13 Serial Peripheral Interface (SPI)

F2802x provides the functions for serial peripheral interface (SPI). By using the SPI blocks in the TI F803x Target library, one can implement the function to communicate with external SPI devices (such as external A/D and D/A converters) easily and conveniently. Writing code manually for SPI devices is often a time-consuming and non-trivial task. With the capability to support SPI, PSIM greatly simplifies and speeds up the coding and hardware implementation process.

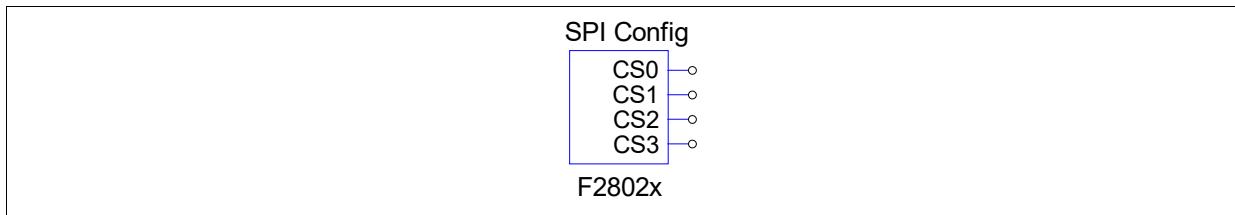
For more detailed descriptions on how to use SPI blocks, please refer to the document "*Tutorial - Using SPI for Real-Time Monitoring.pdf*".

Four SPI function blocks are provided in SimCoder: *SPI Configuration*, *SPI Device*, *SPI Input*, and *SPI Output*, as described below.

9.13.1 SPI Configuration

The SPI Configuration block defines the SPI port, the chip selection pins, and the SPI buffer size. It must be present in a schematic where SPI is used, and this block must be in the main schematic.

Image:

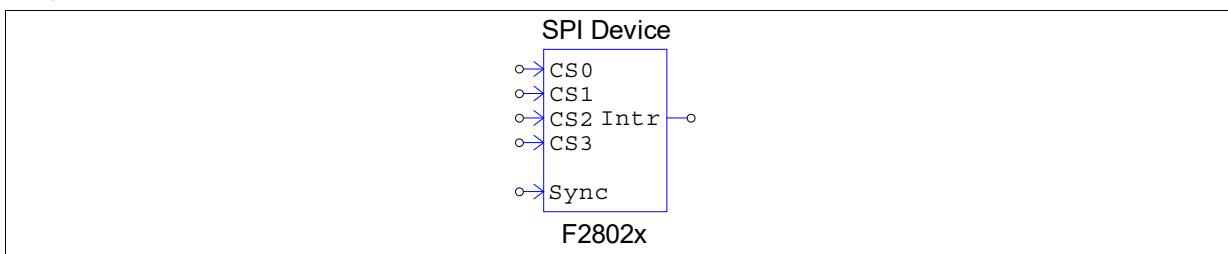


Attributes:

Parameters	Description
SPI Port	Define the SPI port from the options: - SPIA (GPIO 16-19) - SPIA (GPIO 3, 5, 18, 19) - SPIB (GPIO 12-15) - SPIB (GPIO 24-27)
Chip Select Pin0, 1, 2, and 3	The GPIO port of the chip select pin. PSIM supports up to 16 SPI devices, which requires four GPIO pins for chip select, as defined by Chip Select Pin0 to Pin3. These GPIO ports and the SPI slave transmit-enable pin SPISTE are used to generate the chip select signal.
SPI Buffer Size	The buffer size of the SPI commands. Each memory cell of the buffer saves the index of a SPI command. Normally, one can specify the buffer size as 1 plus the number of SPI commands (i.e. Start Conversion Command, Receiving Data Command, Sending Data Command, and Sync. Command) in all SPI Input/Output elements.

9.13.2 SPI Device

The SPI Device block defines the information of the corresponding SPI hardware device. The number of SPI Device blocks in the schematic must be the same as the number of SPI hardware devices.

Image:**Attributes:**

Parameters	Description
Chip Select Pins	The state of the chip select pins corresponding to the SPI device. When the chip select pins are at this state, this SPI device is selected.
Communication Speed (MHz)	SPI communication speed, in MHz.
Clock Type	SPI clock type, as determined by the SPI hardware device. It can be one of the following: - <i>Rising edge without delay</i> : The clock is normally low, and data is latched at the clock rising edge. - <i>Rising edge with delay</i> : The clock is normally low, and data is latched at the clock rising edge with delay. - <i>Falling edge without delay</i> : The clock is normally high, and data is latched at the clock falling edge. - <i>Falling edge with delay</i> : The clock is normally high, and data is latched at the clock falling edge with delay.
Command Word Length	Word length, or the length of the significant bits, of SPI communication commands. It can be from 1 to 16 bits.

Sync. Active Mode	The triggering mode of the synchronization signal of the SPI device. It can be either <i>Rising edge</i> or <i>Falling edge</i> .
SPI Initial Command	The SPI command that initializes the SPI device.
Hardware Interrupt Mode	Specify the type of the interrupt signal that the SPI device generates. This is valid only when the SPI device's interrupt output node is connected to the input of a digital output element. It can be one of the following: <ul style="list-style-type: none"> - <i>No hardware interrupt</i> - <i>Rising edge</i> - <i>Falling edge</i>
Interrupt Timing	Specify how a SPI device generates interrupt when it completes conversion. It can be one of the following: <ul style="list-style-type: none"> - <i>No interrupt</i>: No interrupt is generated. In this case, DSP sends the command to a SPI input device. This device starts the conversion and returns the result in the same command - <i>Multiple interrupt in series</i>: Multiple interrupts are generated in series after each conversion. This is for a SPI device that has one A/D conversion unit and multiple input channels. In this case, DSP send the first conversion command, and the SPI device starts the conversion. When the conversion is complete, the SPI device will generate an interrupt. In the interrupt service routine, DSP will send a command to fetch the conversion result, and start a new conversion of another channel of the same SPI input device. - <i>One-time interrupt</i>: Only one interrupt is generated at the end of the conversion. This is for a SPI device that can perform multiple channel conversions in one request. In this case, DSP sends the command to the SPI input device, and the SPI device completes the conversion of multiple input channels. When all the conversions are complete, the SPI device will generate an interrupt.
Command Gaps (ns)	The gap between two SPI commands, in nsec.
Conversion Sequence	Define the names of the SPI input elements, separated by comma, that determine the conversion sequence. Note that this parameter is valid only when the SPI device generates multiple interrupts in series.

In a schematic, the chip select pins of all the SPI devices are connected to the chip select pins of the SPI Configuration block, without defining how the chip select logic is implemented. In the actual hardware, however, one would need to implement the corresponding chip select logic accordingly.

A SPI command consists of a series of 16-bit numbers separated by comma. In the 16-bit number, only the lower bits are the significant bits used by the command. For example, if the Command Word Length is 8, Bits 0 to 7 are the command, and Bits 8 to 15 are not used.

A SPI device can be either an input device or an output device. For example, an external A/D converter is an input device. Usually DSP will send one or multiple A/D conversion commands to the device, and then set the synchronization signal to start the conversion. The synchronization signal is reset at the next command of the same device.

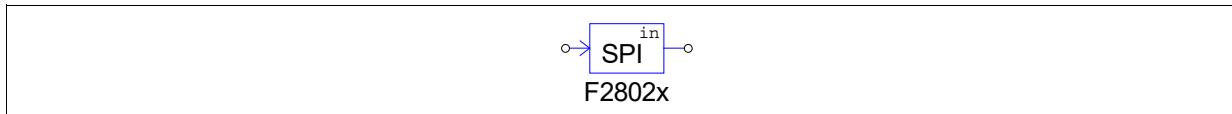
A SPI input device using the synchronization signal usually needs an interrupt pin to trigger DSP to enter the interrupt service routine.

On the other hand, an external D/A converter is an output device. Usually DSP sends one or multiple D/A conversion commands to the device, and then sets the synchronization signal to start the conversion. The synchronization signal is reset at the next command of the same device.

9.13.3 SPI Input

A SPI input device may have multiple input channels. The SPI Input block is used to define the properties of an input channel for SPI communication, and one SPI Input block corresponds to one input channel.

Image:



Attributes:

Parameters	Description
Device Name	Name of the SPI input device.
Start Conversion Command	Command to start conversion, in hex numbers, separated by comma (for example, 0x23,0x43,0x00).
Receiving Data Command	Command to receive data, in hex numbers, separated by comma (for example, 0x23,0x43,0x00).
Data Bit Position	Define where the data bits are in the receiving data string. The format is: $ElementName = \{Xn[MSB..LSB]\}$ where - $ElementName$ is the name of the SPI input device. If it is the current SPI input device, use y instead. - $\{\}$ means that the item in the bracket repeats multiple times. - Xn is the n_{th} word received from the SPI input device, and n start from 0. - $MSB..LSB$ defines the position of the significant bits in the word.
Input Range	Specify the parameter V_{max} that defines the input range. This parameter is valid only when the SPI device is an A/D converter. If the device conversion mode is DC, the input ranges from 0 to V_{max} . If the device conversion mode is AC, the input ranges from $-V_{max}/2$ to $V_{max}/2$.
Scale Factor	Output scale factor K_{scale} . If the scale factor is 0, the SPI device is not an A/D converter, and the result will be exactly the same as what DSP receives from SPI communication. Otherwise, the SPI device is an A/D converter, and the result is scaled based on this factor and the A/D conversion mode.
ADC Mode	The A/D conversion mode of the device. It can be either DC or AC. Note that this parameter is valid only when the device is an A/D converter.
Initial Value	The initial value of the input.

The formula for the *Data Bit Position* defines the data length of a SPI input device. For example, $y=x1[3..0]x2[7..0]$, means that the data length is 12, and the result is the lower 4 bits of the 2nd word and the lower 8 bits of the 3rd word. If the received data string is 0x12,0x78,0xAF, then the result is 0x8AF.

If the scale factor is not 0, the output will be scaled based on the following:

In the DC conversion mode:

- In simulation: $Output = Input \cdot K_{scale}$
- In hardware: $Output = \frac{Result \cdot V_{max} \cdot K_{scale}}{2^{Data_Length}}$

In the AC conversion mode:

- In simulation: $Output = Input \cdot K_{scale}$

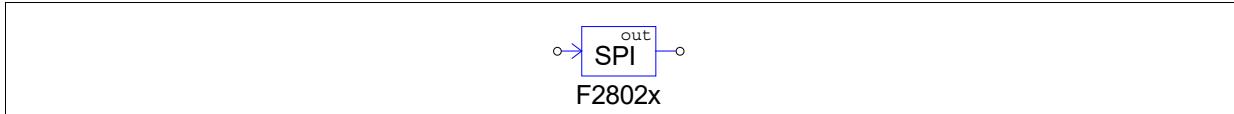
$$\text{- In hardware: } Output = \frac{(Result - 2^{Data_Length-1}) \cdot V_{max} \cdot K_{scale}}{2^{Data_Length-1}}$$

The parameter *Data_Length* is calculated from the Data Bit Position formula.

9.13.4 SPI Output

A SPI output device may have multiple output channels. The SPI Output block is used to define the properties of an output channel for SPI communication, and one SPI Output block corresponds to one output channel.

Image:



Attributes:

Parameters	Description
Device Name	Name of the SPI output device.
Scale Factor	Output scale factor K_{scale} . If the scale factor is 0, the SPI device is not a D/A converter, and the result will be exactly the same as what DSP receives from SPI communication. Otherwise, the SPI device is an D/A converter, and the result is scaled based on this factor and the D/A conversion mode.
Output Range	Specify the parameter V_{max} that defines the output range. This parameter is valid only when the SPI device is an D/A converter. If the device conversion mode is DC, the input ranges from 0 to V_{max} . If the device conversion mode is AC, the input ranges from $-V_{max}/2$ to $V_{max}/2$.
DAC Mode	The D/A conversion mode of the device. It can be either <i>DC</i> or <i>AC</i> . Note that this parameter is valid only when the device is a D/A converter.
Sending Data Command	Command to send the output data, in hex numbers, separated by comma (for example, 0x23,0x43,0x00).
Data Bit Position	Define where the data bits are in the sending data string. The format is: $ElementName = \{Xn[MSB..LSB]\}$ where - <i>ElementName</i> is the name of the SPI output device. If it is the current SPI output device, use <i>y</i> instead. - $\{\}$ means that the item in the bracket repeats multiple times. - <i>Xn</i> is the n_{th} word sent to the SPI output device, and <i>n</i> start from 0. - <i>MSB..LSB</i> defines the position of the significant bits in the word.
Sync. Command	The command to synchronize output channels of the SPI output device, in hex numbers, separated by comma (for example, 0x23,0x43,0x00). This command is used when the SPI output device does not have the synchronization signal.

The formula for the *Data Bit Position* defines the data length of a SPI output device. For example, $y=x1[3..0]x2[7..0]$, means that the data length is 12, and the data is the lower 4 bits of the 2nd word and the lower 8 bits of the 3rd word. If the sending data string is 0x12,0x78,0xAF, then the data is 0x8AF.

If the scale factor is not 0, the output will be scaled based on the following:

In the DC conversion mode:

- In simulation: $Output = Input \cdot K_{scale}$

- In hardware:
$$Output = \frac{Result \cdot K_{scale} \cdot 2^{Data_Length}}{V_{max}}$$

In the AC conversion mode:

- In simulation:
$$Output = Input \cdot K_{scale}$$

- In hardware:
$$Output = 2^{Data_Length} + \frac{Result \cdot K_{scale} \cdot 2^{Data_Length-1}}{V_{max}}$$

The parameter *Data_Length* is calculated from the Data Bit Position formula.

9.14 Interrupt Time

The interrupt time block is used to measure the time interval of an interrupt service routine.

Attributes:

Parameters	Description
Time Output Method	<p>Define how interrupt time is measured. It can be one of the following:</p> <ul style="list-style-type: none"> - <i>SCI (time used)</i>: Using SCI. Time used by the interrupt service routine, in DSP clock count, is measured and is sent out via SCI output. - <i>SCI (time remaining)</i>: Using SCI. Time remaining in the interrupt service routine, in DSP clock count, is measured and is send out via SCI output. The time remaining is defined as the time from the end of the current interrupt to the beginning of the next interrupt. - <i>GPIO0 to GPIO44, or AIO2 to AIO14</i>: Using a GPIO port. A pulse is generated at the specified GPIO port. The pulse is set to high when entering the interrupt, and set to low when exiting the interrupt. An oscilloscope can be used to measure the width of the pulse.
Sampling Frequency	Sampling frequency of the interrupt service routine, in Hz.

When SCI is used, the value is the count of the DSP clock. For example, if the value is 6000, for a 60-MHz DSP clock, the interrupt time will be: $6000 / 60M = 100$ us.

9.15 Project Settings and Memory Allocation

When generating the code for the TI F2802x Hardware Target, SimCoder also creates the complete project files for the TI Code Composer Studio (CCS) development environment, so that the code can be compiled, linked, and uploaded to the DSP.

At the present, CCS version 3.3 is supported. Assuming that the PSIM schematic file is "test.sch", after the code generation, a sub-folder called "test (C code)" will be generated in the directory of the schematic file, and sub-folder will contain the following files:

- test.c Generated C code
- PS_bios.h: Header file for the SimCoder F2802x library
- passwords.asm: File for specifying the DSP code password
- test.pjt: Project file for Code Composer Studio
- F2802x_Headers_nonBIOS.cmd: Peripheral register linker command file
- F28027_FLASH_Lnk.cmd: Flash memory linker command file
- F28027_FLASH_RAM_Lnk.cmd: Flash RAM memory linker command file
- F28027_RAM_Lnk.cmd: RAM memory linker command file

Note: The names of the link command files are assigned with the target hardware if it is not F2802x. For example, if the target hardware is F28027, the file names will be *F28027 FLASH Lnk.cmd*, *F28027 FLASH RAM Lnk.cmd*, and *F28027RAM Lnk.cmd* accordingly.

Besides, the project also needs the following library files:

- PsBiosRamF06xFixpt.lib:SimCoder F2802x library, located in the PSIM folder
- PsBiosRomF06xFixpt.lib:SimCoder F2802x library, located in the PSIM folder
- IQmath.lib: TI's IQmath.lib, located in the PSIM /lib folder
- 2802x_IQmath_BootROMsymbols.libIQmath symbols library, located in the PSIM /lib folder

These library files will be copied automatically to the project folder when the code is generated.

Each time code generation is performed, the .c file and .pj file (in this example, "test.c" and "test.pjt") will be created. If you have made changes manually to these two files, be sure to copy the changed files to a different location. Otherwise the changes will be overwritten when code generation is performed next time.

Project Setting:

In the Code Composer Studio project file, the following settings are provided:

- *RAM Debug*: To compile the code in the debug mode and run it in the RAM memory
- *RAM Release*: To compile the code in the release mode and run it in the RAM memory
- *Flash Release*: To compile the code in the release mode and run it in the flash memory
- *Flash RAM Release*: To compile the code in the release mode and run it in the RAM memory

When RAM Debug or RAM Release is selected, CCS uses the linker command file F2802x_RAM_Lnk.cmd to allocate the program and data space.

When Flash Release is selected, CCS uses the linker command file F2802x_FLASH_Lnk.cmd to allocate the program and data space.

When Flash RAM Release is selected, CCS uses the linker command file F2802x_FLASH_RAM_Lnk.cmd to allocate the program and data space. The memory allocation is the same as in RAM Release.

The code compiled in the release mode is faster than the code in the debug mode. Also, the code in RAM Release or Flash RAM Release is the fastest. The code in RAM Debug is slower, and the code in Flash Release is the slowest. In a development, normally one would start with RAM Debug for easy debugging. Then switch to RAM Release and consequently to Flash Release or Flash RAM Release.

Memory Allocation:

In the generated link files, the memory allocation is defined in the following way.

With the RAM Debug, RAM Release, and Flash RAM Release settings:

RAM Memory 0x0000 - 0x07FF (2K) interrupt vectors stack 0x8000 - 0x9FFF (8K*) program and data space
--

With the Flash Release setting:

RAM Memory 0x0000 - 0x07FF (2K) interrupt vectors stack 0x8000 - 0x9FFF (8K*) data space	Flash Memory 0x3F0000 - 0x3F7FFF (64K**) program password etc.
--	---

Notes:

* The RAM memory predefined by SimCoder for program and data space is:

- For F28027, F28026, F28023, and F28022: From 0x8000 to 0x9FFF (8K)
- For F28021: From 0x8000 to 0x8BFF (6K)
- For F28020 and F280200: From 0x8000 to 0x83FF (2K)

- If the combined program and data space exceeds the size of the RAM space, Flash Release must be selected as the project setting.

** The flash memory predefined by SimCoder for program space is:

- For F28027, F28023, and F28021: From 0x3E8000 to 0x3F7FFF (64K)
- For F28026, F28022, and F28020: From 0x3F4000 to 0x3F7FFF (32K)
- For F280200: From 0x3F6000 to 0x3F7FFF (16K)

10.1 Overview

With the F2837x Hardware Target, SimCoder can generate code that is ready to run on any hardware boards based on Texas Instruments' F2837x fixed-point DSP.

The F2837x Hardware Target will work with all F2837x packages.

The F2837x Hardware Target library includes the following function blocks:

- PWM generators: 3-phase, 2-phase, 1-phase, and APWM
- Variable frequency PWM
- Start/Stop functions for PWM generators
- PWM trip-zone and trip-zone state
- A/D converter
- ADC voltage reference
- Comparator input, output, and DAC
- DAC converter
- Digital input, output, and sample time
- Input X-BAR, output X-BAR, and PWM X-BAR
- Encoder, encoder state, and encoder index/strobe position
- Up/Down counter
- Capture and capture state
- SCI configuration, Input, and output
- SPI configuration, device, input, and output
- CAN configuration, input, output, and remote request
- Interrupt Time
- DSP clock
- Hardware configuration

When generating the code for a system that has multiple sampling rates, SimCoder will use the interrupts of the PWM generators for the PWM sampling rates. For other sampling rates in the control system, it will use the Timer 1 interrupt first, and then Timer 2 interrupt if needed. If there are more than three sampling rates in the control system, the corresponding interrupt routines will be handled in the main program by software.

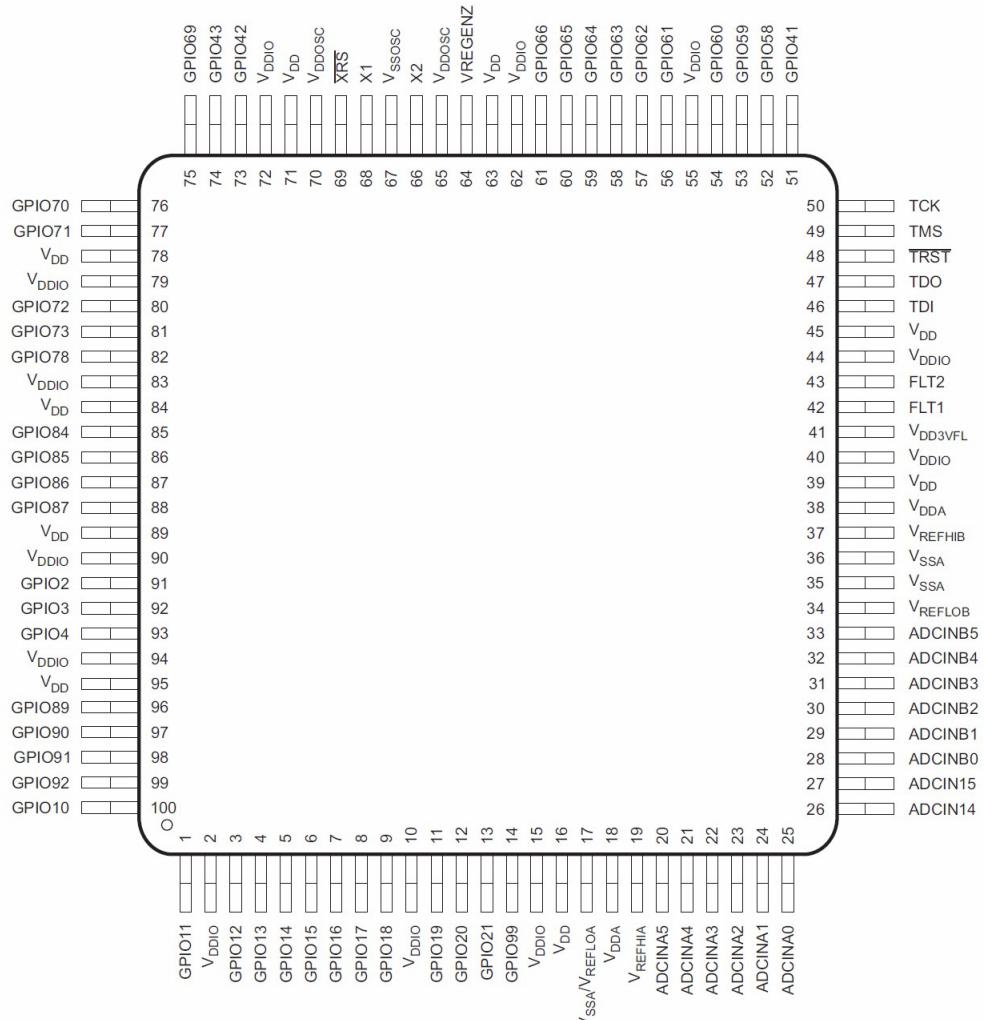
In TI F2837x, PWM generators can generate hardware interrupt. SimCoder will search and group all the elements that are connected to the PWM generator and have the same sampling rate as the PWM generator. These elements will be automatically placed and implemented in an interrupt service routine in the generated code.

In addition, digital input, encoder, capture, and trip-zone can also generate hardware interrupt. Each hardware interrupt must be associated with an interrupt block (described in Section 4.5 of this Manual), and each interrupt block must be associated with an interrupt service routine (a subcircuit that represents the interrupt service routine). For example, if a PWM generator and a digital input both generate interrupt, there should be one interrupt block and one interrupt service routine for each of them.

The definitions of the elements in the F2837x Hardware Target library are described in this Chapter.

The figure below shows the F2837x 100-pin PZP port assignment. Only the GPIO functions are show on GPIO pins. The other port functions will be explained in next section about hardware configuration.

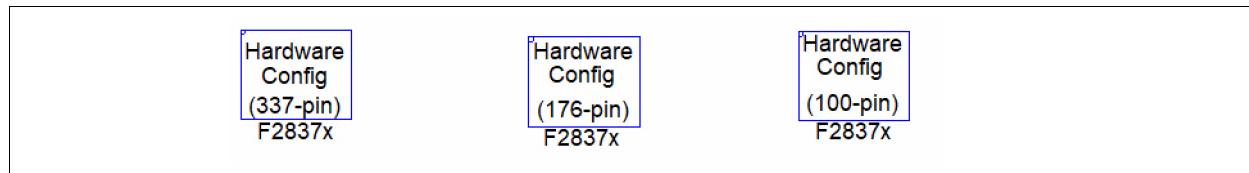
F28004x 100-Pin PZP Port Assignment



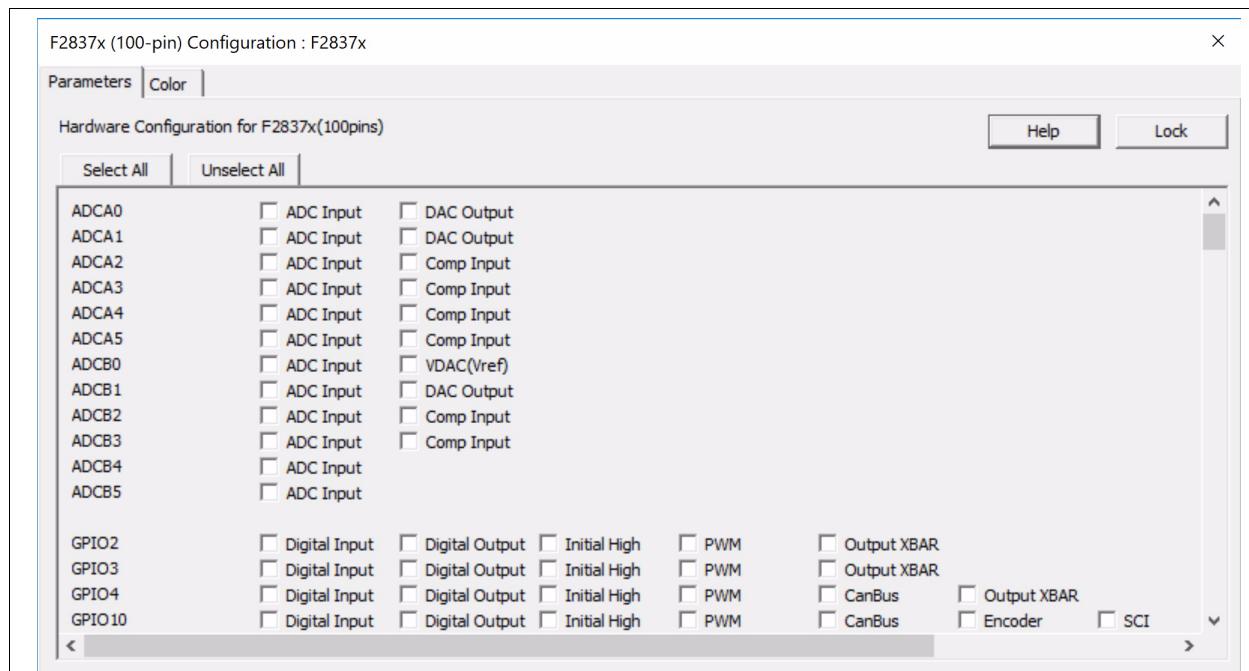
10.2 Hardware Configuration

F2837x provides analog channels and individually programmable multiplexed GPIO ports. Many of the GPIO ports can perform one of several functions. For example, port GPIO0 can be used either as a digital input, a digital output, or a PWM output. Some of the A/D channels can also be defined as comparator input. Therefore, user must assign the functions to the GPIO ports correctly according to the PSIM circuit schematic.

Image:



The dialog window of the block is shown below:



The Hardware Configuration block is for user to specify the I/O ports usage of the F2837x hardware. Every port in use must be assigned correctly. The ports not in use can be left unchecked.

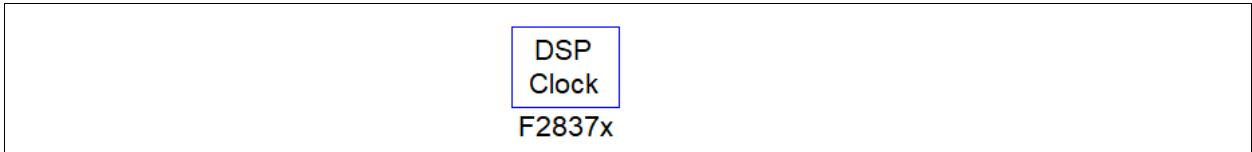
To make changes, click on the button **Unlock**. After changes are made, click on **Lock** to lock the configuration.

For each GPIO port, a check box is provided for each of its available function. When a box is checked, the GPIO port is configured for that particular function. Only one box can be checked for each GPIO port. For example, if the checkbox for "Digital Input" is checked for port GPIO1, this port is configured as a digital input, and hence, cannot be used for any other functions. If it is used as a PWM output in the PSIM circuit schematic, an error message will be generated.

10.3 DSP Clock

The DSP Configuration block defines the DSP clock source and frequency, as well as the DSP speed.

Image:



Attributes:

Parameters	Description
DSP Clock Source	There are three ways of providing system clock to F2837x: <ul style="list-style-type: none">- Internal oscillator 1- Internal oscillator 2- External oscillator
External Clock (MHz)	Frequency of the external clock on the DSP board, in MHz. The frequency must be an integer, and the maximum frequency allowed is 20 MHz. This parameter is ignored if the DSP clock source is selected to be internal oscillator 1 or 2.
DSP Speed (MHz)	DSP Speed, in MHz. The speed must be an integer, and must be an integer multiple of the external clock frequency. The maximum DSP speed allowed is 194MHz when using internal oscillators, and is 200MHz when using external oscillator.

If a DSP Configuration block is not used in a schematic, the default values of the DSP Configuration block will be used:

- External oscillator
- 20MHz
- 200MHz

10.4 PWM Generators

F2837x contains 7 sets of PWM modules for 100pin package, 12 sets for 176pin package and 330pin packages. Each set of PWM module has two output ports.

In SimCoder, the PWM's can be used in the following ways:

- **1-phase PWM generators:** with two outputs complementary to each other.
- **1-phase PWM generators with phase shift:** with two outputs complementary to each other.
- **2-phase PWM generators:** with the two outputs of each PWM generator in special operation mode, maybe not in a complementary way.
- **3-phase PWM generators:** consisting three 1-phase PWM generators of consecutive order, such as PWM 123 (consisting of PWM 1, 2, and 3) and PWM 456 (consisting of PWM 4, 5, and 6);

These PWM generators can trigger the A/D converter, and use trip-zone signals.

Beside the PWM generators described above, there are also 6 single output APWM generators that use the same resources as the captures. These APWM generators have restricted functionality comparing to the PWM generators. They can not trigger the A/D converter and can not use trip-zone signals. Also, because of the common resources, when a particular port is used for the capture, it can not be used for the PWM generator.

Note that all the PWM generators in SimCoder include one switching period delay internally. That is, the input value of a PWM generator is delayed by one cycle before it is used to update the PWM output. This delay is needed to simulate the delay inherent in the DSP hardware implementation.

PWM generators have a parameter called "PWM Freq. Scaling Factor". It can be set to 1 to 15. The hardware limit is 3. If the scaling factor is greater than 3, PWM will use an unused PWM to generator interrupt at the sampling frequency. This unused PWM is only used to generate a periodic interrupt, and its outputs can still be

used for other functions. If there is no unused PWM in the system, a timer will be used.

PWM generators can generate an interrupt in two ways:

- Periodical interrupt. The interrupt frequency is the same as the sampling frequency. It can be generated in the following ways:
 - If PWM is selected as the interrupt source, interrupt will be generated by PWM itself at the start of the PWM carrier wave.
 - If the A/D converter is selected as the interrupt source, PWM will trigger the A/D converter to start the conversion. After the A/D conversion is complete, interrupt will be generated.
- Trip-Zone interrupt. 3 tripzone singles (TZ1, TZ2 and TZ3) from input X-bar and 4 DC trip events (or their combination) can be used as interrupt source. If the trip-zone signal of the PWM is active, a trip-zone interrupt will be generated. Before entering the trip-zone interrupt, all PWM outputs will be set according to the Trip Action definition. DC trip events are set at the corresponding PWM trip-zone element.

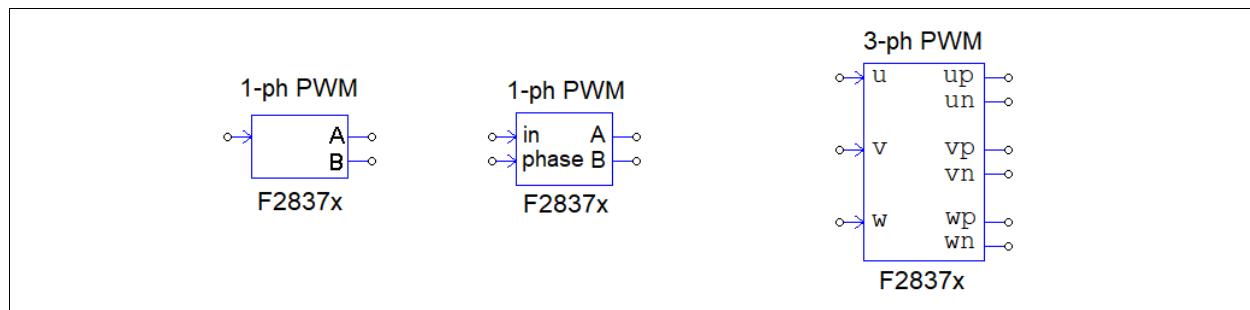
10.4.1 3-Phase PWM, 1-phase PWM, and 1-phase PWM (phase shift)

The 3-phase PWM generator consists of three 1-phase PWM blocks of consecutive order. Therefore, the attributes for **3-phase PWM** block, **1-Phase PWM** block and **1-phase PWM (phase shift)** block are mostly the same.

The difference between the 1-Phase PWM and 1-phase-PWM (phase shift) blocks is how the phase shift is defined. The 1-phase PWM block defines the phase shift through a parameter which is a constant, while the 1-Phase PWM (phase shift) block reads the phase shift from an external input (labeled as "phase" in the image). The phase shift is in degree.

In the 3-phase PWM generator image, "u", "v", and "w" refer to the three phases (alternatively they are called phase "a", "b", and "c"). The letter "p" refers to the positive output, and "n" refers to the negative output. For example, for 3-phase PWM 123, "up" is PWM1A, and "un" is PWM1B.

Image:



Attributes:

Parameters	Description
PWM Source	Source of the PWM generator. Each 1-phase PWM block has two outputs (PWM A and PWM B) assigned to two GPIO pins. Each 3-phase PWM module consists of three 1-phase PWM blocks. For example, 3-phase <i>PWM 123 (GPIO0)</i> consists of PWM1, PWM2, and PWM3. Therefore, each 3-phase PWM has 6 GPIO pins.
Output Mode (Not used for 3-phase)	Output mode of the PWM generator. It can be one of the following: <ul style="list-style-type: none"> - <i>Use PWM A&B</i>: Both PWM outputs A and B are used, and they are complementary. - <i>Use PWM A</i>: Only PWM output A is used. - <i>Use PWM B</i>: Only PWM output B is used. - <i>No PWM output</i>: No PWM output is used (As a timer).

Dead Band Type	The type of PWM dead band: <ul style="list-style-type: none"> - Active high complementary - Active low complementary - Active high - Active low.
Rising Edge Dead Time (us)	The dead time for the rising edge of the PWM generator, in us.
Falling Edge Dead Time (us)	The dead time for the falling edge of the PWM generator, in us.
Sampling Frequency	Sampling frequency of the PWM generator, in Hz. The calculation is done and the PWM signal duty cycle is updated based on this frequency.
PWM Freq. Scaling Factor	The ratio between the PWM switching frequency and the sampling frequency. It can be from 1 to 15. For example, if the sampling frequency is 50 kHz and the scaling factor is 3, the PWM switching frequency will be 150 kHz. That is switches will operate at 150 kHz. But gating signals will be updated at 50 kHz, or once per 3 switching cycles.
Carrier Wave Type	Carrier wave type and the initial PWM output state. It can be one of the following: <ul style="list-style-type: none"> - <i>Triangular (start low)</i>: Triangular wave, and the initial PWM output state is low. - <i>Triangular (start high)</i>: Triangular wave, and the initial output state is high. - <i>Sawtooth (start low)</i>: Sawtooth wave, and the initial output state is low. - <i>Sawtooth (start high)</i>: Sawtooth wave, with the initial output state is high.
Trigger ADC	Setting whether for the PWM generator to trigger the A/D converter. It can be one of the followings: <ul style="list-style-type: none"> - <i>Do not trigger ADC</i>: PWM does not trigger the A/D converter. - <i>Trigger ADC</i>: PWM will trigger A/D converter.
ADC Trigger Position	A/D trigger position ranges from 0 to a value less than 1. When it is 0, the A/D converter is triggered at the beginning of the PWM cycle, and when it is 0.5, the A/D converter is triggered at the 180° position of the PWM cycle.
Use Trip-Zone 1, 2, 3	Define whether the PWM generator uses the trip-zone signal or not. It can be one of the followings: <ul style="list-style-type: none"> - <i>Do not use</i>: Disable the i_{th} trip-zone signal. - <i>One shot</i>: The PWM generator uses the trip-zone signal in the one-shot mode. Once triggered, the PWM must be started manually. - <i>Cycle by cycle</i>: The PWM generator uses the trip-zone signal in the cycle-by-cycle basis. The trip-zone signal is effective within the current cycle, and PWM will automatically re-start in the next cycle.
Select Trip Events	Specify the combination of DC trip events applied to the current PWM. It can be one or any combination of DCAEVT1, DCAEVT2, DCBEVT1, and DCBEVT2.
Trip Action	Define how the PWM generator responds to the trip action. It can be one of the followings: <ul style="list-style-type: none"> - <i>High impedance</i>: PWM outputs in high impedance - <i>PWM A high & B low</i>: Set PWM A high and B low. - <i>PWM A low & B high</i>: Set PWM A low and B high. - <i>PWM A both high</i>: Set both PWM A and B high (not used for 3-phase). - <i>PWM A both low</i>: Set both PWM A and B low (not used for 3-phase). - <i>No action</i>: No action taken.
Peak-to-Peak Value	Peak-to-peak value V_{pp} of the carrier wave

Offset Value	DC offset value V_{offset} of the carrier wave
Phase Shift (for 1-phase only)	Phase shift of the output with respect to the reference PWM generator output, in deg. For example, when the phase shift is -30, the output will be shifted to the right (lagging) by 30 deg. with respect to the reference PWM output.
Initial Input Value	Initial value of the PWM block. For three initial input values are set for 3-phase PWM block's three inputs u , v , and w .
Use HRPWM (Not used for 3-phase)	Define the high-resolution PWM. It can be one of the followings: <ul style="list-style-type: none"> - <i>Do not use HRPWM</i>: Do not use high-resolution PWM - <i>Use HRPWM without calibration</i>: Use high-resolution PWM without calibration - <i>Use HRPWM with calibration</i>: Use high-resolution PWM with calibration
Start PWM at Beginning	When it is set to <i>Start</i> , PWM will start right from the beginning. If it is set to <i>Do not start</i> , one needs to start PWM using the Start PWM block.
Simulation Output Mode	The simulation output mode can be set to either of the followings: <ul style="list-style-type: none"> - <i>Switching mode</i>: the outputs of the PWM block are PWM signals. - <i>Average mode</i>: the outputs of the PWM block are average mode signals. <p>In the average mode, if the carrier wave is from negative to positive, and the absolute values of the negative peak and the positive peak are equal (for example, the carrier wave is from -1 to +1, or from -5 to +5), the modulation is considered as an ac signal modulation. Otherwise, the modulation is considered as a dc signal modulation. For example, modulation in a 3-phase or single-phase inverter is an ac modulation, and modulation in a buck converter is a dc modulation.</p> <p>In the ac signal modulation, if the input u of the PWM block is V_u, the output up and un in average mode will be:</p> $V_{up} = V_u / (V_{pp} + V_{offset})$ $V_{un} = -V_{up}$ <p>In this case, V_u is between $-(V_{pp} + V_{offset})$ and $(V_{pp} + V_{offset})$, and V_{up} is between -1 to +1.</p> <p>In the dc signal modulation, the output up and un in average mode will be:</p> $V_{up} = (V_u - V_{offset}) / V_{pp}$ $V_{un} = 1 - V_{up}$ <p>In this case, V_u is between V_{offset} and $V_{pp} + V_{offset}$, and V_{up} is between 0 to +1.</p> <p>When it is set to the average mode, the PWM block outputs can be connected to a converter/inverter in the average mode model.</p>

Phase Shift

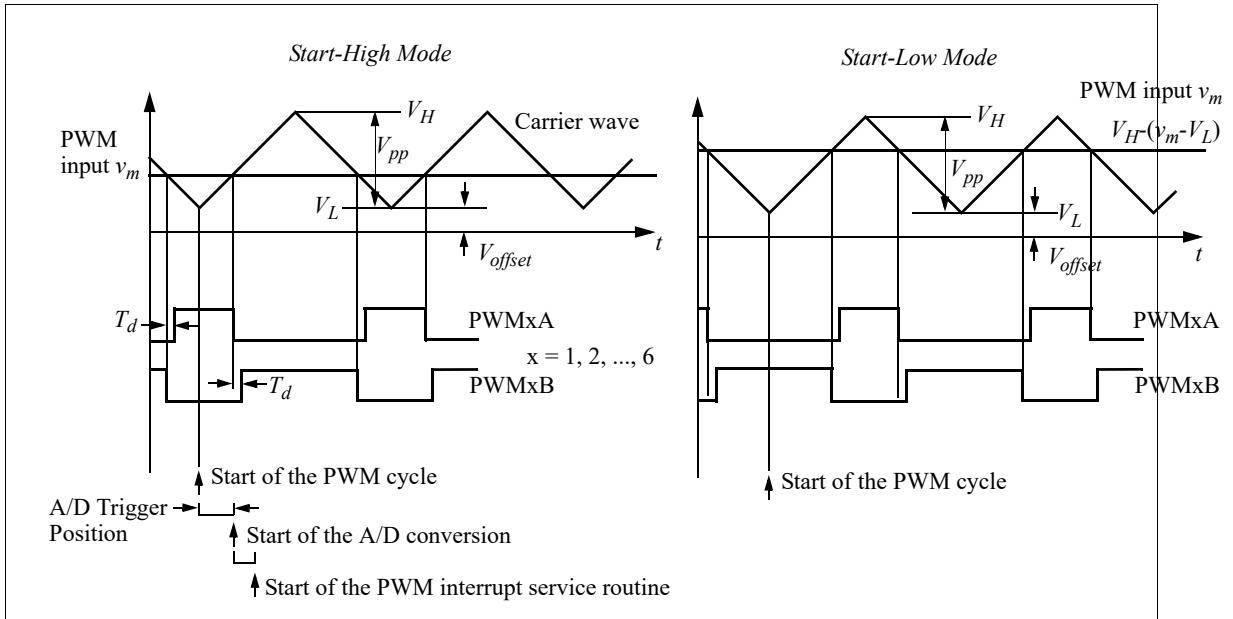
A 1-phase PWM generator can generate PWM signal that is phase shifted with respect to another PWM signal. The way how PWM blocks are defined for phase shift is explained in Section 8.4.5.

The phase shift value is in degrees. When the value is -30°, the output will be shifted to the right (lagging) by 30° of the switching cycle with respect to the reference PWM generator output. This is equivalent to shifting the PWM carrier wave to the right by 30°. When the phase value is 30°, the output will be shifted to the left (leading) by 30°.

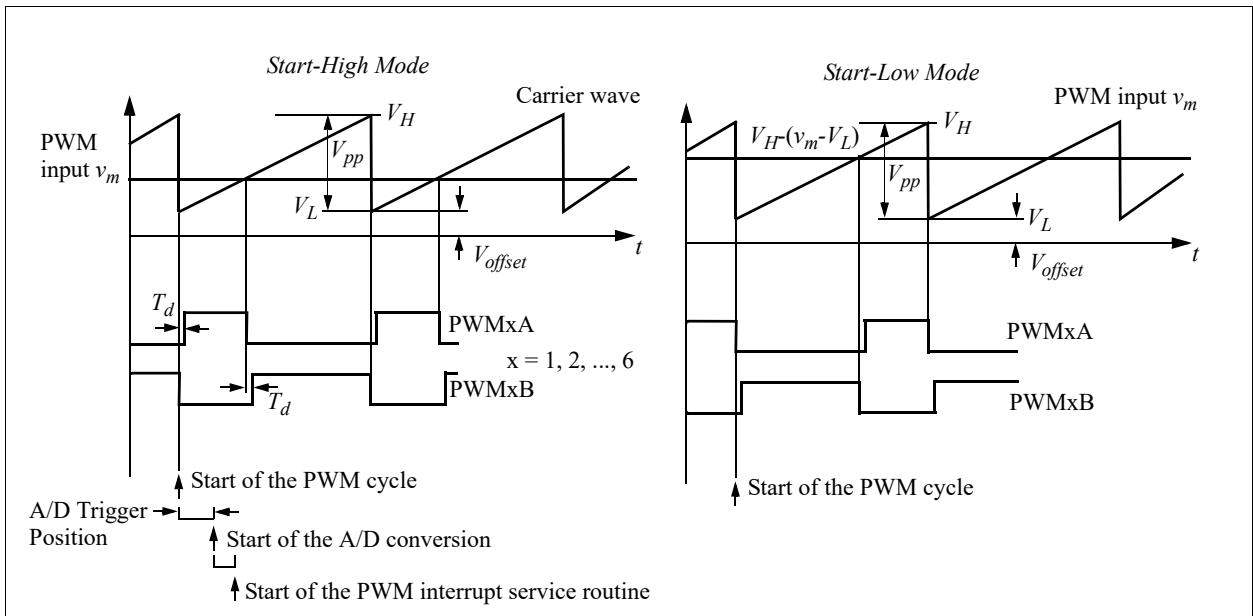
Carrier Wave

There are two types of carrier waveforms: triangular wave (with equal rising and falling slope intervals) and sawtooth wave. In addition, there are two operation modes: start-low and start-high modes, as explained below.

The input and output waveforms of a PWM generator with the triangular carrier wave are shown below:



The input and output waveforms of a PWM generator with the sawtooth carrier wave are shown below:



The figures above show how the dead time is defined, and the time sequence when the PWM generator triggers the A/D converter. If triggering the A/D converter is selected, from the start of the PWM cycle, after a certain delay defined by the A/D trigger position, the A/D conversion will start. After the A/D conversion is completed, the PWM interrupt service routine will start.

If the PWM generator does not trigger the A/D converter, the PWM interrupt service routine will start at the beginning of the PWM cycle.

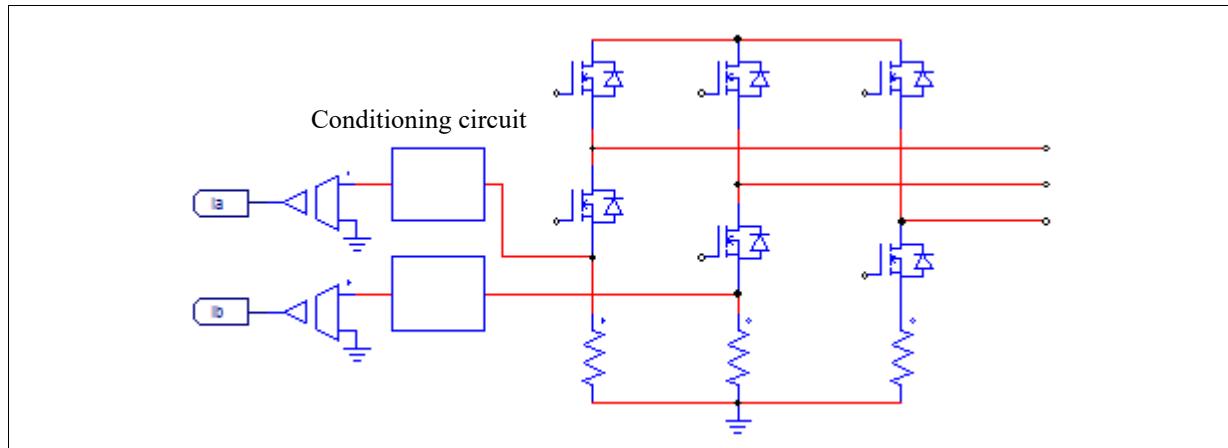
The figures above also show how the start-high and start-low modes work. Assume that the PWM input is v_m , and the lowest value of the carrier wave is V_L and the highest value is V_H . In the start-high mode, the PWM positive output PWMA is high at the beginning of the switching cycle, and it remains high as long as the input v_m is greater than the carrier wave. For example, for a carrier wave from 0 to 1, $V_L=0$, and $V_H=1$. If $v_m=0.2$, the PWM output PWMA will remain high as long as the carrier is less than 0.2.

On the other hand, in the start-low mode, the PWM positive output PWMA is low at the beginning of the

switching cycle, and it is high when the carrier wave is greater than the value $V_H(v_m - V_L)$. For example, for a carrier wave from 0 to 1, $V_L=0$, and $V_H=1$. If $v_m=0.2$, the PWM output PWMA will be high as long as the carrier is greater than 0.8.

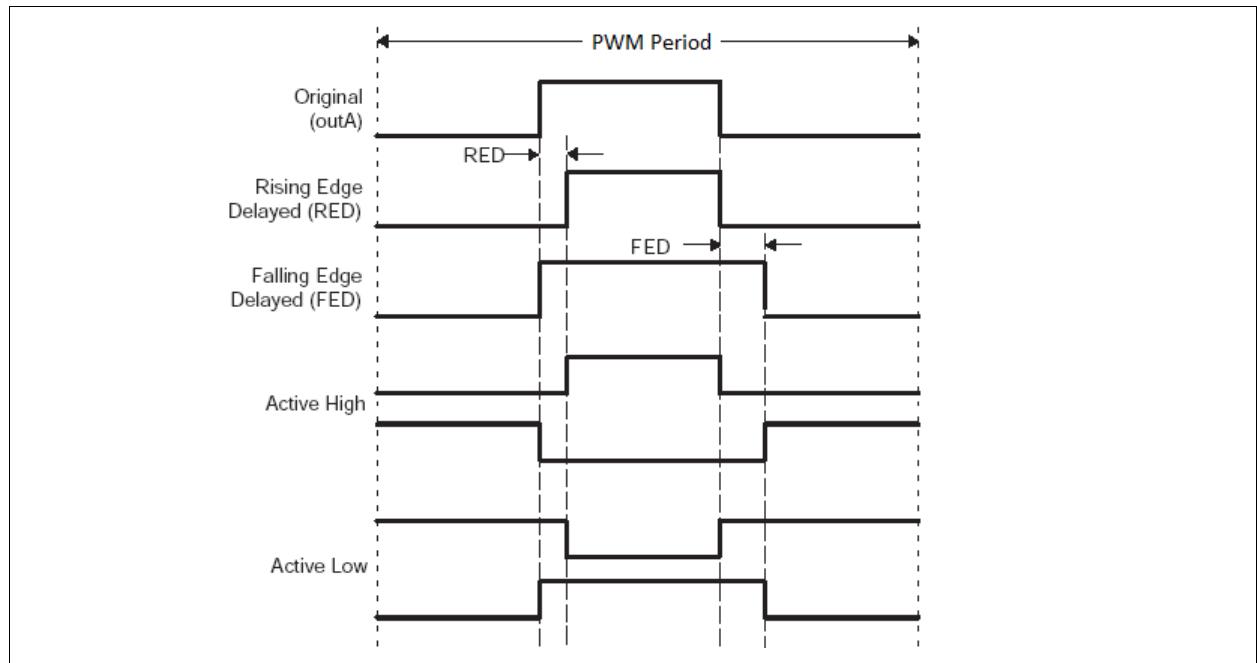
The carrier start mode depends on how switch currents are measured. In a 3-phase inverter, for example, if top switch currents are measured, the start-high mode should be selected. This ensures that at the beginning of the cycle, the top switch gating signal is high and the current is conducting. On the other hand, if bottom switch currents are measured, the start-low mode should be selected. This ensures that at the beginning of the cycle, the top switch gating signal is low, and the bottom switch gating signal is high and the current is conducting.

For example, in the circuit below, the bottom switch currents of Phase A and B are measured. In this case, the carrier start-low mode should be selected.



Note: In the start-low mode, the PWM input v_m is converted to $V_H(v_m - V_L)$ internally before it is compared with the carrier wave to generate the PWM signal. With the conversion, both the start-low and start-high modes will have the same duty cycle expression. For example, for a sawtooth wave with $V_L=0$ and $V_H=1$, or for a triangular wave with $V_L= -V_H$, the duty cycle D of the PWMA output in both the start-low and start-high modes is: $D = v_m/V_H$.

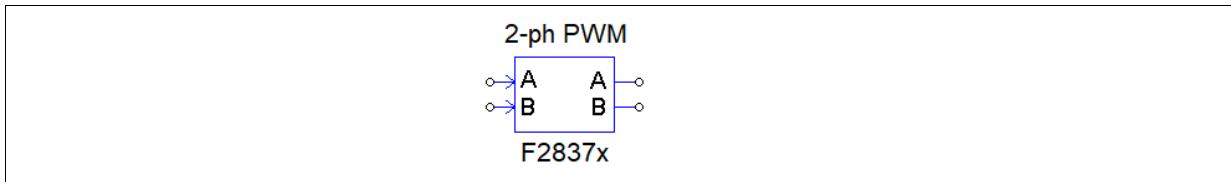
The PWM dead band acts as below.



10.4.2 2-Phase PWM Generator

A 2-phase PWM generator has two inputs and two outputs, and operates in one of the six pre-defined operation modes.

Image:



Attributes:

Parameters	Description
PWM Source	Source of the PWM generator. It can be PWM 1 to PWM 8.
Dead Band Type	The type of PWM dead band, it can be either Active high or Active low.
Sampling Frequency	Sampling frequency of the PWM generator, in Hz. The calculation is done and the PWM signal duty cycle is updated based on this frequency.
PWM Freq. Scaling Factor	The ratio between the PWM switching frequency and the sampling frequency. It can be from 1 to 100. For example, if the sampling frequency is 50 kHz and the scaling factor is 3, the PWM switching frequency will be 150 kHz. That is switches will operate at 150 kHz. But gating signals will be updated at 50 kHz, or once per 3 switching cycles.
Carrier Wave Type	Carrier wave type and the initial PWM output state. It can be one of the following: - <i>Sawtooth</i> : Sawtooth wave. - <i>Triangular</i> : Triangular wave.

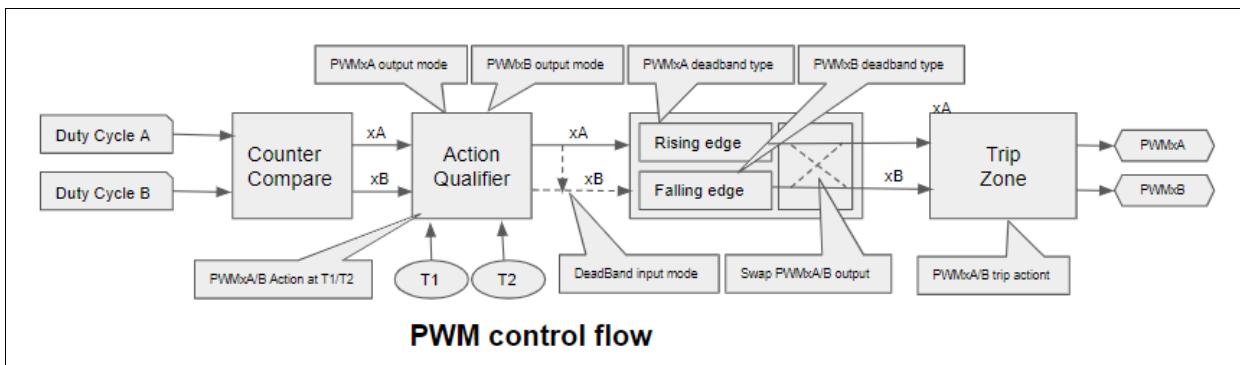
PWMxA Output Mode PWMxB Output Mode	<p>PWMxA and PWMxB output mode can be one of the followings:</p> <ul style="list-style-type: none"> - <i>Do nothing</i>: PWM output is not affected by PWM duty cycle value. - <i>Toggle at valley</i>: PWM output toggles at the beginning of PWM period. - <i>Toggle at peak</i>: PWM output toggles at the peak of PWM carrier wave. - <i>Toggle at peak/valley</i>: PWM output toggles at both the peak/valley of PWM carrier wave. - <i>Set/Reset at valley/peak</i>: PWM output set to high at the valley, and reset to low at the peak of the carrier wave - <i>Reset/Set at valley/peak</i>: PWM output reset to low at the valley, and set to high at the peak of the carrier wave. - <i>Set at valley</i>: PWM output set to high at the beginning of PWM period. - <i>Reset at valley</i>: PWM output reset to low at the beginning of PWM period. - <i>Set at peak</i>: PWM output set to high at the peak of the carrier wave. - <i>Reset at peak</i>: PWM output reset to low at the peak of the carrier wave. <p>The followings are for PWMxA only:</p> <ul style="list-style-type: none"> - <i>Set high at CMPA</i>: PWM output starts low at the beginning of PWM period and turns high when CMPA is set. - <i>Set low at CMPA</i>: PWM output starts high at the beginning of PWM period and turns low when CMPA is set. - <i>Set high/low at CMPA/B</i>: PWM output starts low at the beginning of PWM period and turns high when CMPA is set; turns low when CMPB is set. - <i>Set low/high at CMPA/B</i>: PWM output starts high at the beginning of PWM period and turns low when CMPA is set; turns high when CMPB is set. <p>The followings are for PWMxB only:</p> <ul style="list-style-type: none"> - <i>Complementary to PWMxA</i>: PWM output is complementary to PWMxA. - <i>Set high at CMPB</i>: PWM output starts low at the beginning of PWM period and turns high when CMPA is set. For PWMxA only. - <i>Set low at CMPB</i>: PWM output starts high at the beginning of PWM period and turns low when CMPA is set. For PWMxA only.
Trigger ADC	Setting whether for the PWM generator to trigger the A/D converter. It can be one of the following: <ul style="list-style-type: none"> - <i>Do not trigger ADC</i>: PWM does not trigger the A/D converter. - <i>Trigger ADC</i>: PWM will trigger the A/D converter.
ADC Trigger Position	A/D trigger position ranges from 0 to a value less than 1. When it is 0, the A/D converter is triggered at the beginning of the PWM cycle, and when it is 0.5, the A/D converter is triggered at the 180° position of the PWM cycle.
T1 Source T2 Source	T1 and T2 event sources for PWM trip actions, can be one of the followings: <ul style="list-style-type: none"> - <i>Do not use</i>, - <i>DCAEVT1</i>, - <i>DCAEVT2</i>, - <i>DCBEVT1</i>, - <i>DCBEVT2</i>, - <i>TZ1</i>, - <i>TZ2</i>, - <i>TZ3</i>.

PWMxA Action at T1(Up)	PWMxA (PWMxB) output can be forced at the following actions when T1 (T2) event occurs at PWM waveform up (down) count period: <ul style="list-style-type: none"> - <i>Do nothing</i>, - <i>Clear</i>, - <i>Set</i>, - <i>Toggle</i>.
PWMxA Action at T1(Down)	
PWMxB Action at T1(Up)	
PWMxB Action at T1(Down)	
PWMxA Action at T2(Up)	
PWMxA Action at T2(Down)	
PWMxB Action at T2(Up)	
PWMxB Action at T2(Down)	
Dead Band Input Mode	Specify the dead band's input mode: <ul style="list-style-type: none"> - <i>Use both inputs</i>: PWM deadband sub-module's inputs are from both PWMxA and PWMxB - <i>Use PWMxA input only</i>: PWM dead band sub-module's inputs are from PWMxA only
PWMxA Dead Band Type	The type of PWM dead band, it can be one of the followings: <ul style="list-style-type: none"> - <i>No Delay</i>: No delay applied - <i>Rising/Falling edge delay</i>: Apply rising/falling edge delay to PWMxA/B - <i>Rising/Falling edge delay (invert)</i>: Apply rising/falling edge delay to PWMxA/B then invert it.
PWMxB Dead Band Type	
Swap PWMxA/B Output	Specify if PWM output A and B should be swapped
Rising edge Dead Time (us)	The dead time for the rising edge of the PWM generator, in us
Falling Edge Dead Time (us)	The dead time for the falling edge of the PWM generator, in us
Use Trip-Zone 1, 2, 3	Define whether the PWM generator uses the trip-zone signal or not. It can be one of the followings: <ul style="list-style-type: none"> - <i>Do not use</i>: Disable the i_{th} trip-zone signal. - <i>One shot</i>: The PWM generator uses the trip-zone signal in the one-shot mode. Once triggered, the PWM must be started manually. - <i>Cycle by cycle</i>: The PWM generator uses the trip-zone signal in the cycle-by-cycle basis. The trip-zone signal is effective within the current cycle, and PWM will automatically re-start in the next cycle.
Select Trip Events	Specify the combination of DC trip events applied to the current PWM. It can be one or any combination of DCAEVT1, DCAEVT2, DCBEVT1, and DCBEVT2.
Cycle-by-cycle Lock Clear	Specify when to clear the lock when in cycle-by-cycle mode: <ul style="list-style-type: none"> - <i>At PWM valley</i> - <i>At PWM valley/peak</i>.

PWMxA Trip Action(Up)	Define how the PWM generator responds to the trip action. It can be one of the following: <ul style="list-style-type: none"> - <i>High impedance</i>: PWM outputs in high impedance - <i>Force high</i>: Set PWM A high and B low. - <i>Force low</i>: Set PWM A low and B high. - <i>Toggle</i>: Toggle the output - <i>No action</i>: No action taken.
PWMxA Trip Action(Down)	
PWMxB Trip Action(Up)	
PWMxB Trip Action(Down)	
Peak Value	Peak value V_{pk} of the carrier wave. The lower end is 0, not $-V_{pk}$
Phase Shift	Phase shift of the output with respect to the reference PWM generator output, in deg. For example, when the phase shift is -30, the output will be shifted to the right (lagging) by 30 deg. with respect the reference PWM output.
Initial Input Value A	Initial value of the inputs A and B.
Initial Input Value B	
Use HRPWM	Define the high-resolution PWM. It can be one of the followings: <ul style="list-style-type: none"> - <i>Do not use HRPWM</i>: Do not use high-resolution PWM - <i>Use HRPWM without calibration</i>: Use high-resolution PWM without calibration - <i>Use HRPWM with calibration</i>: Use high-resolution PWM with calibration
Start PWM at Beginning	When it is set to "Start", PWM will start right from the beginning. If it is set to "Do not start", one needs to start PWM using the "Start PWM" function block.

The 2-phase PWM generator generates two PWM signals with specific configurations for typical power converter applications. The PWM carrier wave is either sawtooth or triangular, it ranges from 0 to the peak value V_{pk} .

PWM module's **control flow** is as follows:



T1/T2 events are defined at F2837x Trip Zone block.

The following PWMxA/B **output mode combination** is allowed in SimCoder:

PWMxA output mode	PWMxB output mode	Sawtooth	Triangular
Toggle	Toggle		
Toggle	Set high at duty cycle 2	PwmxB set high at duty cycle 2	PwmxB set high/low at count up/down period at duty cycle 2.

Toggle	Set low at duty cycle 2	PwmxB sets high at duty cycle 2	PwmxB set low/high at count up/down period at duty cycle 2.
Set high at duty cycle 1	Toggle	PwmxA sets high at duty cycle 1	PwmxA sets high/low at count up/down period at duty cycle 1
Set high at duty cycle 1	Set high at duty cycle 2	PwmxA/B set high at duty cycle 1/2	PwmxA set high/low at count up/down period at duty cycle 1; PwmxB set high/low at count up/down period at duty cycle 2.
Set high at duty cycle 1	Set low at duty cycle 2	PwmxA/B set high/low at duty cycle 1/2	PwmxA set high/low at count up/down period at duty cycle 1; PwmxB set low/high at count up/down period at duty cycle 2.
Set low at duty cycle 1	Set high at duty cycle 2	PwmxA/B sets low/high at duty cycle 1/2	PwmxA sets low/high at count up/down period at duty cycle 1; PwmxB set high/low at count up/down period at duty cycle 2.
Set low at duty cycle 1	Set low at duty cycle 2	PwmxA/B sets low at duty cycle 1/2	PwmxA sets low/high at count up/down period at duty cycle 1; PwmxB set low/high at count up/down period at duty cycle 2.
Set high/low at duty cycle 1/2	Toggle	PwmxA sets high/low at duty cycle 1/2	PwmxA sets high/low at count up/down period at duty cycle 1/2.
Set low/high at duty cycle 1/2	Toggle	PwmxA sets low/high at duty cycle 1/2	PwmxA sets high/low at count up/down period at duty cycle 1/2.

"Toggle" in the above table can be one of following modes: Do nothing, Toggle at valley, Toggle at peak, Toggle at valley/peak, Set/reset at valley/peak, Reset/set at valley/peak, Set at valley, Reset at valley, Set at peak and Reset at peak.

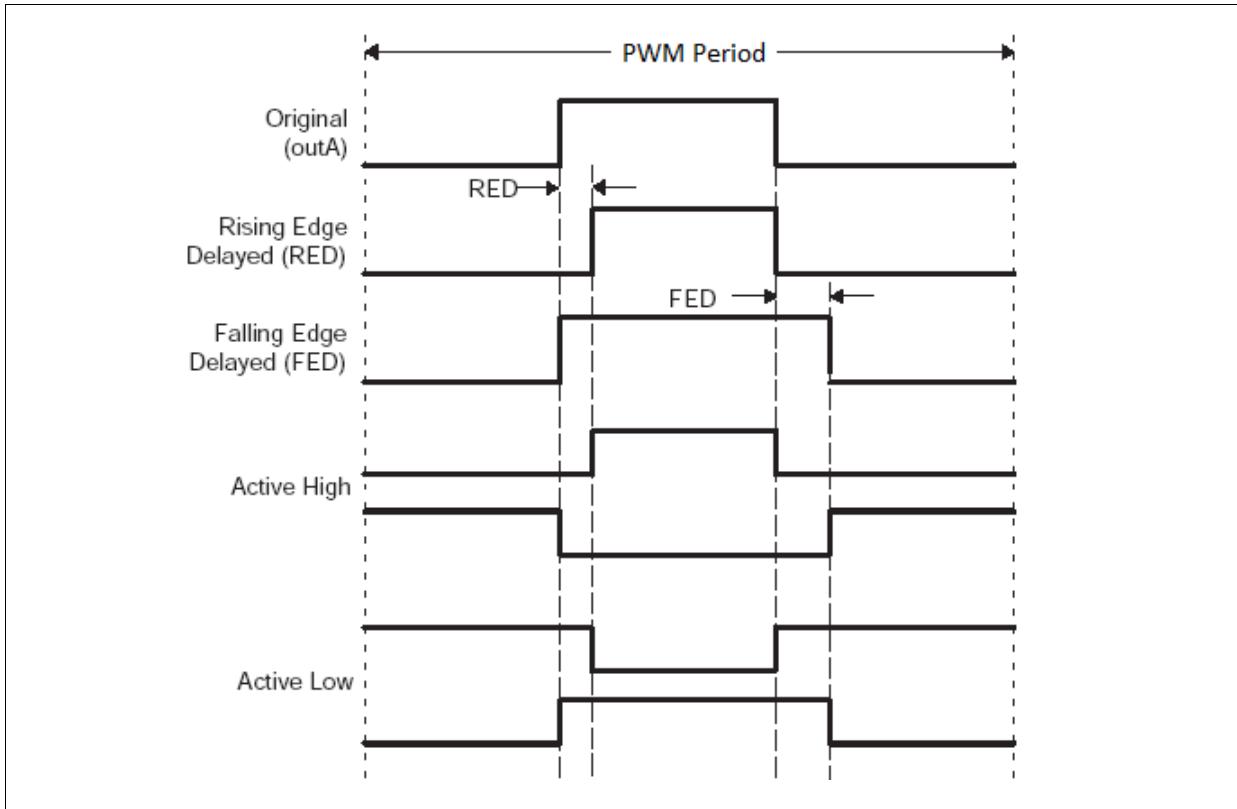
Dead band can be allowed only when PWMxB output mode is *Complementary to PWMxA*. Usually rising edge delay is set to PWMxA, falling edge delay is set to PWMxB, if rising edge delay is set to PWMxB and/or falling edge is set to PWMxA, Set parameter 'Swap PWMxA/B Output' to 'Yes'.

Dead band only applies to PWM actions specified at the following parameters:

PWMxA/B Output Mode, PWMxA/B Action at T1/2(Up/Down).

Dead band is not applied to PWM trip actions.

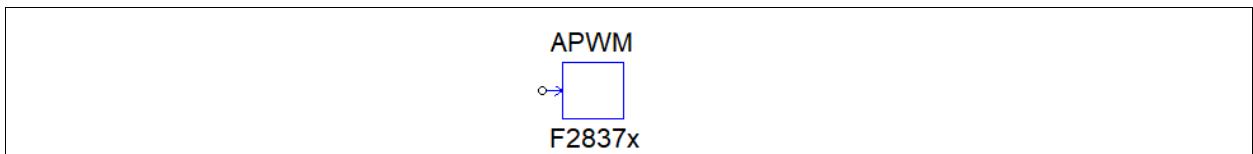
Dead band setting acts as the graph below.



10.4.3 Single PWM (shared with capture)

A single PWM (shared with capture) generator, also called APWM, shares the same resource as captures. It has restricted functionality as compared to other PWM generators. They cannot trigger the A/D converter and cannot use trip-zone signals. Also, because of the common resource, when a shared GPIO port is used for the capture, it can not be used for the PWM generator.

Image:



Attributes:

Parameters	Description
PWM Source	Source of the PWM generator. It can be any from APWM1 to APWM6
PWM Frequency	Frequency of the PWM generator, in Hz
Carrier Wave Type	The carrier wave type and the initial PWM output state. It can be one of the following: - <i>Sawtooth (start low)</i> : Sawtooth wave, with the PWM output in the low state initially. - <i>Sawtooth (start high)</i> : Sawtooth wave, with the PWM output in the high state initially.

Stop Action	The output status when the PWM generator is stopped. It can be one of the following: - <i>Output low</i> : The PWM output will be set to low. - <i>Output high</i> : The PWM output will be set to high.
Peak-to-Peak Value	Peak-to-peak value of the carrier wave
Offset Value	DC offset value of the carrier wave
Phase Shift	Phase shift of the output with respect to the reference PWM generator, in deg.
Initial Input Value	Initial value of the input
Start PWM at Beginning	When it is set to <i>Start</i> , PWM will start right from the beginning. If it is set to <i>Do not start</i> , one needs to start PWM using the Start PWM block.

Similar to 1-phase PWM generators, an APWM generator can generate a PWM signal that has a phase shift with respect to another PWM generator. The way how PWM blocks are defined for phase shift is explained in Section 10.4.4.

10.4.4 Synchronization Between PWM Blocks

Three types of PWM blocks can be synchronized, and phase shifts can be defined between each other: **1-phase PWM**, **1-phase PWM (phase shift)**, and **Single PWM (shared with capture) (APWM)**.

A 1-phase PWM block can generate PWM signal that is phase shifted with respect to another PWM signal. There is one series in regular PWM blocks: PWM 1, 2, 3, 4, 5, 6, 7, and 8.

Similarly, there is one series in APWM blocks for phase shift: PWM 1, APWM 1, 2, and 3.

The definitions of the PWM blocks for phase shift are described below.

- The reference PWM and the PWM being phase shifted must be from the same series. That is, PWM 1 can be the reference, and PWM 2, 3, and etc. can be phase shifted with respect to PWM 1. Or PWM 2 can be the reference, and PWM 3, 4, and etc. can be phase shifted with respect to PWM 2.

This also applies to APWM blocks. For example, PWM 1 can be the reference, and APWM 1, 2, and etc. can be phase shifted. Or APWM 1 can be the reference, and APWM 2, 3, and etc. can be phase shifted.

- The reference PWM and the PWM being shifted must be consecutive in the series, unless the skipped PWM is not used. For example, if PWM 1, 2, and 3 are all used, but PWM 2 is not synchronized with PWM 1 and 3, this is not allowed. However, if PWM 2 is not used in the circuit, it is ok to use PWM 1 as the reference and phase shift PWM 3.

For example, the following definitions are correct, with the first PWM as the reference and the subsequent PWM blocks phase shifted:

PWM 1 (reference), PWM 2, PWM 3, PWM 4, PWM 5, PWM 6, PWM 7

PWM 2 (reference), PWM 3

PWM 4 (reference), PWM 5

PWM 5 (reference), PWM 6

PWM 6 (reference), PWM 7

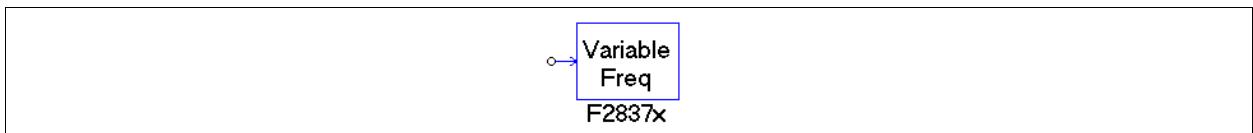
PWM 1 (reference), PWM 2, PWM 3, PWM 4, PWM 5, PWM 6, APWM 1, APWM 2, APWM 3

PWM 1 (reference), APWM 1, APWM 2, APWM 3

10.5 Variable Frequency PWM

The Variable Frequency PWM block provides the function to change the sampling frequency of a PWM generator. The image and parameters are shown below.

Image:



Attributes:

Parameters	Description
PWM Source	Source of the PWM generator. It can be one listed in the pull-down menu.
Adjust Interrupt Pos.	Specify if the interrupt position is adjusted with the frequency. It can be one of the following: - <i>Do not adjust</i> : The interrupt position will remain unchanged as calculated with the base frequency. - <i>Adjust</i> : The interrupt position will be recalculated at the beginning of each cycle based on the new frequency.
Adjust Ramp Compensation	Specify if the ramp compensation of the comparator DAC block is adjusted with the frequency. It can be one of the following: - <i>Do not adjust</i> : The ramp compensation will remain unchanged as calculated with the base frequency. - <i>Adjust</i> : The ramp compensation will be recalculated at the beginning of each cycle based on the new frequency.

The sampling frequency of the corresponding PWM block will be changed at the beginning of the next PWM period as follows:

$$\text{PWM_Frequency} = \text{PWM_Base_Frequency} / \text{Input_Value}$$

where *PWM_Base_Frequency* is the sampling frequency of the corresponding PWM block, and *Input_Value* is the input value of this block.

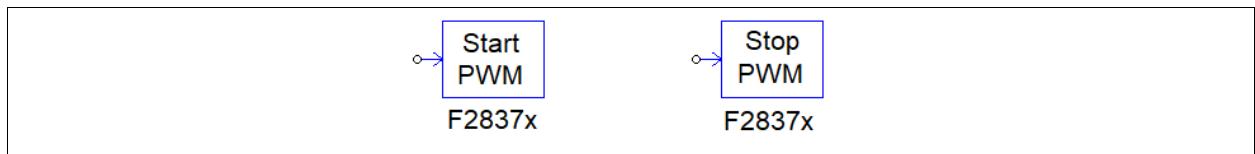
If the interrupt position is to be adjusted, the interrupt position will be recalculated in each cycle. Since adjusting the interrupt position takes time, if the frequency change is small, it is recommended not to adjust the interrupt position.

Similarly, if the ramp compensation is to be adjusted, the ramp compensation will be recalculated in each cycle. Since adjusting the ramp compensation takes time, if the frequency change is small, it is recommended not to adjust the ramp compensation.

10.6 Start PWM and Stop PWM

The Start PWM and Stop PWM blocks provide the function to start/stop a PWM generator. The images and parameters are shown below.

Image:



Attributes:

Parameters	Description
PWM Source	The source of the PWM generator. It can be any of PWM blocks or captures in the pull-down list

10.7 PWM Trip-Zone And Trip-Zone State

All the F2837x's PWM DC trip-zone signals come from the PWM X-Bar, They are TRIP1(TZ1), TRIP2(TZ2), TRIP3(TZ3), TRIP4, TRIP5, TRIP7, TRIP8, TRIP9, TRIP10, TRIP11 and TRIP12. There is no TRIP6.

Trip-zone is used to handle external fault or trip conditions. The corresponding PWM outputs can be programmed to respond accordingly.

The Trip-Zone State element indicates the type of the trip-zone signal, whether it is one-shot or cycle-by-cycle.

10.7.1 PWM Trip-Zone

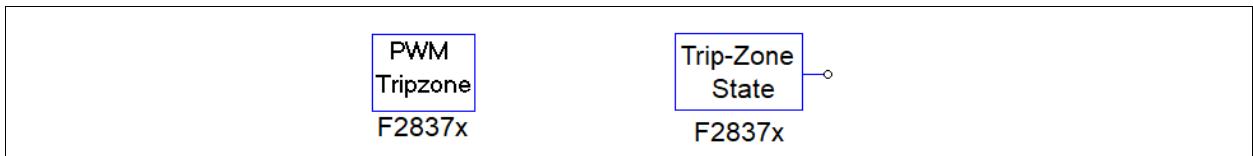
Any PWM X-Bar signal can be used in one or multiple PWMs. The interrupt generated by trip-zone signals are handled by the interrupt block.

Any trip-zone signals though Digital Compare trigger a trip action in the specified active level (high or low).

The trip-zone interrupt can be generated in either one-shot mode or cycle-by-cycle mode, as defined in the PWM generator parameter input. In the cycle-by-cycle mode, the interrupt only affects the PWM output within the current PWM cycle. On the other hand, in the one-shot mode, interrupt will set the PWM output permanently, and the PWM generator must be restarted to resume the operation.

Valley switch is supported. When simulating valley switching circuit, one should set a proper simulation step according to the specified trip-zone signals, usually a smaller time step is recommended. If trip-zone signal changes multiple times in one time step, SimCoder could not simulate the circuit correctly.

Image:



Attributes for Trip-Zone:

Parameters	Description
PWM Source	Source of the PWM generator. It can be any of the PWM blocks in the pull-down list.
DC Trip Src1(DCAH)	Digital compare (DC) trip source DCAH for PWMxA. The PWM channel may have up to two DC trip sources: DCAH and DCAL. The trip source can be one of the following: <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>TZi(Ti)</i>: PWM uses trip-zone <i>i</i> as the digital compare input signal, where <i>i</i>=1, 2, 3. - <i>TRIPi(Ti)</i>: PWM uses TRIP<i>i</i> as the digital compare input signal, where <i>i</i>=4, 5, 7 - 12. - <i>Other combinations</i>: PWM uses one of the trip signal combinations. If the PWM source is a 3-phase PWM generator, PWMA refers to the outputs "up", "vp", and "wp" for the 3 top switches.
PWMA DC Trip Src2(DCAL)	Digital compare (DC) trip source DCAL for PWMxA. The trip source can be one of the following: <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>TZi(Ti)</i>: PWM uses trip-zone <i>i</i> as the digital compare input signal, where <i>i</i>=1, 2, 3. - <i>TRIPi(Ti)</i>: PWM uses TRIP<i>i</i> as the digital compare input signal, where <i>i</i>=4, 5, 7 - 12. - <i>Other combinations</i>: PWM uses one of the trip signal combinations.
PWMA 1-shot Evt (DCAEVT1)	Define how the one-shot signal is used for the DC trip signal of PWMA. The active level of the DC trip signal can be selected from the following: <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>Trip source 1 is low</i>: PWM is tripped if the source signal 1 is low. - <i>Trip source 1 is high</i>: PWM is tripped if the source signal 1 is high. - <i>Trip source 2 is low</i>: PWM is tripped if the source signal 2 is low. - <i>Trip source 2 is high</i>: PWM is tripped if the source signal 2 is high. - <i>Trip source 1 low & source 2 high</i>: PWM is tripped if the source 1 signal is low and at the same time source 2 signal is high.

PWMA CBC Evt (DCAEVT2)	Define how the cycle-by-cycle signal is used for the DC trip signal of PWMA. The active level of the DC trip signal can be selected from the following: <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>Trip source 1 is low</i>: PWM is tripped if the source signal 1 is low. - <i>Trip source 1 is high</i>: PWM is tripped if the source signal 1 is high. - <i>Trip source 2 is low</i>: PWM is tripped if the source signal 2 is low. - <i>Trip source 2 is high</i>: PWM is tripped if the source signal 2 is high. - <i>Trip source 1 low & source 2 high</i>: PWM is tripped if the source 1 signal is low and at the same time source 2 signal is high.
DC Trip Src1(DCBH)	Digital compare (DC) trip source DCBH for PWMxB. The PWM channel may have up to two DC trip sources: DCBH and DCBL. The trip source can be one of the following: <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>TZ_i(T_i)</i>: PWM uses trip-zone <i>i</i> as the digital compare input signal, where <i>i</i>=1, 2, 3. - <i>TRIP_i(T_i)</i>: PWM uses TRIP_i as the digital compare input signal, where <i>i</i>=4, 5, 7 - 12. - <i>Other combinations</i>: PWM uses one of the trip signal combinations. <p>If the PWM source is a 3-phase PWM generator, PWMxB refers to the outputs "un", "vn", and "wn" for the 3 bottom switches.</p>
PWMB DC Trip Src2(DCBL)	Digital compare (DC) trip source DCBL for PWMxB. The trip source can be one of the following: <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>TZ_i(T_i)</i>: PWM uses trip-zone <i>i</i> as the digital compare input signal, where <i>i</i>=1, 2, 3. - <i>TRIP_i(T_i)</i>: PWM uses TRIP_i as the digital compare input signal, where <i>i</i>=4, 5, 7 - 12. - <i>Other combinations</i>: PWM uses one of the trip signal combinations.
PWMB 1-shot Evt (DCBEVT1)	Define how the one-shot signal is used for the DC trip signal of PWMB. The active level of the DC trip signal can be selected from the following: <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>Trip source 1 is low</i>: PWM is tripped if the source signal 1 is low. - <i>Trip source 1 is high</i>: PWM is tripped if the source signal 1 is high. - <i>Trip source 2 is low</i>: PWM is tripped if the source signal 2 is low. - <i>Trip source 2 is high</i>: PWM is tripped if the source signal 2 is high. - <i>Trip source 1 low & source 2 high</i>: PWM is tripped if the source 1 signal is low and at the same time source 2 signal is high.
PWMA CBC Evt (DCBEVT2)	Define how the cycle-by-cycle signal is used for the DC trip signal of PWMB. The active level of the DC trip signal can be selected from the following: <ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>Trip source 1 is low</i>: PWM is tripped if the source signal 1 is low. - <i>Trip source 1 is high</i>: PWM is tripped if the source signal 1 is high. - <i>Trip source 2 is low</i>: PWM is tripped if the source signal 2 is low. - <i>Trip source 2 is high</i>: PWM is tripped if the source signal 2 is high. - <i>Trip source 1 low & source 2 high</i>: PWM is tripped if the source 1 signal is low and at the same time source 2 signal is high.

DC Event Filter Source	Source of the digital compare event filter. It can be one of the following: <ul style="list-style-type: none"> - <i>Do not use</i>: Do not use DC event filter. - <i>DCAEVT1</i>: Use DCAEVT1 as the filter source. - <i>DCAEVT2</i>: Use DCAEVT2 as the filter source. - <i>DCBEVT1</i>: Use DCBEVT1 as the filter source. - <i>DCBEVT2</i>: Use DCBEVT2 as the filter source.
Blanking Window Pos (us)	Blanking window start position in a PWM period, in us.
Blanking Window Width (us)	Width of the blanking window, in us. The width is limited by the hardware, and can be calculated as below: $255 / \text{CPU Frequency}$ <p>For example, when the CPU speed is 90MHz, the width range is 0 to 2.83us.</p>
Blanking Window Range	Specify how the blanking window is applied. It can be one of the following: <ul style="list-style-type: none"> - <i>In the window</i>: The blanking action is applied with the window defined (from the start position for a window width defined) - <i>Out of window</i>: The blanking action is applied outside the window defined.
Applying Event Filtering	Specify how event filtering is applied to digital compare events. The event filtering can be applied to any combinations of the following digital compare events: DCAEVT1, DCAEVT2, DCBEVT1, and DCBEVT2
Enable Valley Switching	Specify the DC event that is used for valley switch signal, one of the following DC events can be chose: DCAEVT1, DCAEVT2, DCBEVT1, and DCBEVT2
Toggle Event Selection	Specify the chosen trigger event, PWM starts to count up when this trigger event meets the edge specified by " Trigger Event Edge Type "
Toggle Event Type	Specify trigger event edge type. It can be one of the followings: rising edge, falling edge and rising/falling edge.
Toggle Event Count	Specify trigger event edge count, PWM starts to count filter event edge when this trigger event counter equals to this value.
Filter Event Start Edge	Specify the start edge count for the specified filter event. Filter event is chose by " DC Event Filter Source ".
Filter Event Stop Edge	Specify the stop edge count for the specified filter event. <i>Event Start Edge <= Event Stop Edge</i> .

Software DelayTime (us)	Specify software delay time, in us.
Delay Calculate Mode	<p>Specify delay time calculate mode. When the condition specified by parameter 'Trigger Event Edge Count' meets, the specified filter event passes through after the delay time decided by this parameter.</p> <p>Valley delay below is the interval between Filter Event Start Edge and Filter Event Stop Edge.</p> <p>The calculation mode of delay time can be one of the follows:</p> <ul style="list-style-type: none"> - No delay time: Delay time is 0. - Use software delay: Delay time is software delay time. - Use Software delay + valley delay: Delay time is software delay time + valley delay. - Use Software delay + valley delay/2: Delay time is software delay time + valley delay / 2. - Use Software delay + valley delay/4: Delay time is software delay time + valley delay / 4. - Use Software delay + valley delay/16: Delay time is software delay time + valley delay / 16.

Note: when defining the interrupt block associate with trip-zone, the "Device Name" parameter of the interrupt block should be the name of the PWM generator, not the trip-zone block name. For example, if a PWM generator called "PWM_G1" uses trip-zone 1 in the trip-zone block "TZ1". The "Device Name" of the corresponding interrupt block should be "PWM_G1", not "TZ1". The "Channel Number" parameter in the interrupt block is not used in this case.

The trip-zone interrupt can be generated in either one-shot mode or cycle-by-cycle mode, as defined in the PWM generator parameter input. In the cycle-by-cycle mode, the interrupt only affects the PWM output within the current PWM cycle. On the other hand, in the one-shot mode, interrupt triggers a trip action when the input signal is low (0). will set the PWM output permanently, and the PWM generator must be restarted to resume the operation.

10.7.2 Trip-Zone State

The Trip-Zone State element indicates whether the trip-zone signal is in one-shot mode or cycle-by-cycle mode when it triggers a PWM generator to generate an interrupt.

Attributes for Trip-Zone State:

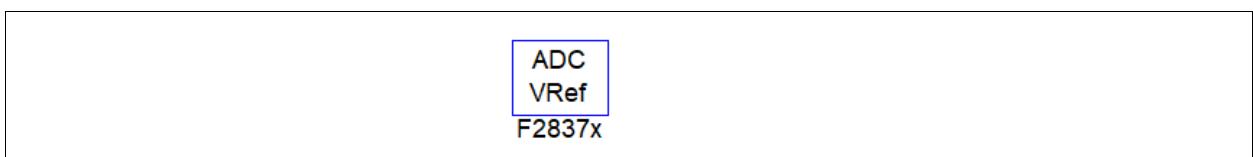
Parameters	Description
PWM Source	Source of the PWM generator. It can be any one in the pull-down list

The trip-zone state element is usually used in the trip-zone interrupt routine to indicate the operation mode of the trip-zone signal. When the output of this element is 1, it means that the trip-zone signal is in one-shot mode. When the output is 0, the trip-zone signal is in cycle-by-cycle mode.

10.8 ADC Voltage Reference

The Voltage Reference for ADCs, DACs and comparators.

Image:



Attributes:

Parameters	Description
ADC- x High Voltage Ref	Specify high voltage reference for ADC block x , where x is A, B, C, and D.
ADC- x Low Voltage Ref	Specify low voltage reference for ADC block x , where x is A, B, C, and D. This value must be set to 0.
Use ADC-B0 as VDAC	Specify if ADC-B0 is used as VDAC.
DAC Voltage Ref	Specify the voltage reference value of VDAC.

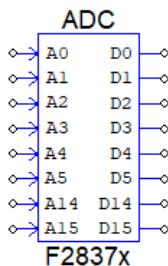
All high voltage reference (include VDAC) can be set in the range of 2.4V to 3.3V. All low voltage reference must be set to 0.

If ADC-B0 is used as VDAC, the DACs and Comparator DACs can use VDAC as voltage reference; otherwise VDAC can not be selected in the DACs and Comparator DACs parameter dialogs. PSIM does not need ADC-B0 to connect a voltage reference in schematic, but in a target system, ADC-B0 must connect to a voltage reference.

10.9 A/D Converter

F2837x target provides four of 8-channel single-ended 12-bit A/D converters: ADC-A, ADC-B, ADC-C. and ADC-D.

The image and parameters of the A/D converter in the SimCoder library for F2837x target are described below.

Image:**Attributes:**

Parameters	Description
ADC Source	Specify ADC source, it can be one of the followings: ADC-A, ADC-B, ADC-C and ADC-D.
Chn i Gain	Gain of the i_{th} A/D converter channel, where i ranges from 0 to 5, 14, and 15.
Chn i Sample Time (us)	Specify the sample time of the A/D converter channel i , where i ranges from 0 to 5, 14 and 15.

Conversion Order	Order of the A/D conversion. If the field is left blank (undefined), the conversion will be done based on the sequential numbers of the A/D channel. ADC channel name used in this parameter is as follows: A0, A1, A2, A3, A4, A5, A14, A15. For example, if A0, A2, A4, A14, and A15 are used, the conversion will be done in this order: A0, A2, A4, A14, and A15. If you wish certain channels to be performed first, you can define the order here. For example, if the conversion order is defined as: A4,A0,A2,A14,A15 The conversion will be done in the order defined, that is, A4 first, then A0, then A2, and so on.
High Priority PIE Selection	Specify if the highest priority interrupt uses <i>PIE Group1</i> or <i>PIE Group10</i> .

All four ADC converters have Chn14 and Chn15, but these two channels are the same inputs. Therefore, they can be used only in one ADC converter.

The output of the A/D converter is scaled based on: $V_{out} = \text{Gain} * V_{in}$, where V_{in} is the value at the input port of the A/D converter.

Each ADC converter has its own voltage reference, these voltage references are specified in an *ADC Voltage Reference block*. High voltage reference can be set from 2.4V to 3.3V, low voltage reference must be set to 0V.

An A/D converter can be triggered by one or more of the PWM generators, by Timer1 or by Timer2. In a schematic, if an A/D channel is not associated with a PWM, one should connect a ZOH block at the output of the A/D channel so that SimCoder will use the selected timer as trigger source.

Each ADC converter has up to 4 interrupts, one is in PIE Group1 with the highest interrupt priority, other 3 are in PIE Group10 with lower interrupt priority.

If any sampling rate uses any ADC channels, SimCoder assigns an ADC interrupt for the sampling rate. If setting parameter 'High Priority PIE Selection' to PIE group 1, SimCoder assigns the highest sampling rate to PIE1 ADC interrupt, the other lower sampling rates use PIE group 10; If setting parameter 'High Priority PIE Selection' to PIE group 10, SimCoder assigns the all sampling rates related to this ADC converter use PIE group 10.

If the same sampling rate uses in multiple ADC converters, SimCoder checks the total conversion time on this sampling rate for all ADC converters, and sets the ADC converter with the longest conversion time to cause interrupt, the other ADC converters do not need to cause interrupt. In this way all related conversions are done

10.10 Comparator

In F2837x, there are 8 comparators. They share the same ports with ADC input channels. PSIM will report an error if a port is defined as a comparator input but is used as a A/D converter input.

10.10.1 Comparator Input

In F2837x, there are 8 comparators. Their inputs share the same ADC ports with ADC input channels as shown below:

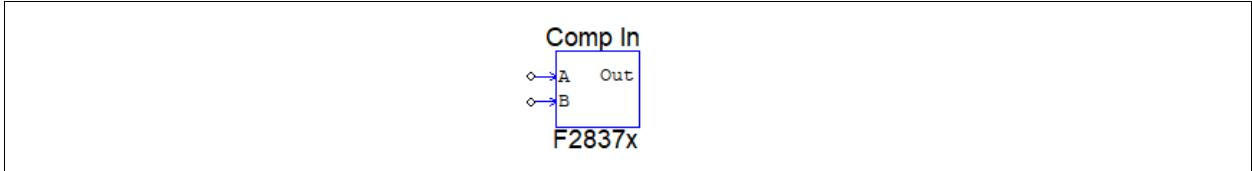
Comparator Number and Input Name	Port Assigned
Comparator1 Input A and B	ADCA2, ADCA3
Comparator2 Input A and B	ADCA4, ADCA5
Comparator3 Input A and B	ADCB2, ADCAB3
Comparator4 Input A and B	ADCC2, ADCC3
Comparator5 Input A and B	ADCC4, ADCC5
Comparator6 Input A and B	ADCD0, ADCD1

Comparator7 Input A and B	ADCD2, ADCD3
Comparator8 Input A and B	ADC14, ADC15

Only one function can be designated for each port. Simcoder will report error if a port is defined as comparator input but is also used as a A/D converter channel in the same PSIM circuit schematic.

Each comparator has 2 side comparators: high side and low side. For each side comparator, Input A is always from an external analog input, side comparator's another input or the output of inner DAC. If Input B is not used in the current element, it can be used as an ADC input.

Image:



Attributes

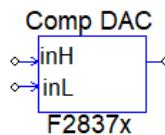
Parameters	Description
Comparator Source	Specify Comparator source, it can be one of the followings: Comparator1, Comparator2, ..., Comparator8.
PWM Sync. Source	Specify the synchronization PWM of this Comparator, it can be <i>No PWM sync. used</i> or one of the PWM generators in the pull-down list
PWM's Sync. Pos (us)	Specify PWM synchronization position of this Comparator, in us. The comparator uses the latest input value for Comparator DAC when it receives the synchronization signal.
Comparator Usage	Specify which signal is used. it can be one of the followings: <ul style="list-style-type: none"> - <i>Use TRIPH signal</i>: Only TRIPH (TRIPOUTH) has been used. - <i>Use TRIPL signal</i>: Only TRIPL (TRIPOUTL) has been used. - <i>Use TRIPH/TRIPL signal</i>: Both TRIPH (TRIPOUTH) and TRIPL (TRIPOUTL) have been used.
Input Hysteresis Selection	Specify the type of hysteresis applied for input signal. This information is ignored in simulation. It can be one of the followings: <ul style="list-style-type: none"> - <i>None</i>: No hysteresis applied. - <i>Typical hysteresis</i>: Typical hysteresis applied. - <i>2x hysteresis</i>: 2 times of hysteresis applied. - <i>3x hysteresis</i>: 3 times of hysteresis applied. - <i>4x hysteresis</i>: 4 times of hysteresis applied
High Output Invert	Specify if inverting the output of high side comparator.
High Output Sync.	Specify if synchronization mode of the output of high side comparator. It can be one of the followings: <ul style="list-style-type: none"> - <i>Asynchronous</i>: No synchronization applied. - <i>Sync. by SysClk</i>: Input signal is synchronized by system clock. - <i>Digital filter result</i>: Input signal is filtered. - <i>Latched result</i>: Input signal is filtered and is latched when it becomes active.
High Filter Width (us)	Specify the filter window width for high side comparator, in us. This information is valid only when 'High Output Sync.' is selected as 'Digital filter result' or 'Latched result'.

High Threshold (%)	Specify the filter threshold for high side comparator, in %. This information is valid only when 'High Output Sync.' is selected as 'Digital filter result' or 'Latched result'.
Low Output Invert	Specify if inverting the output of low side comparator.
Low Output Sync.	Specify if synchronization mode of the output of low side comparator. It can be one of the followings: <ul style="list-style-type: none"> - <i>Asynchronous</i>: No synchronization applied. - <i>Sync. by SysClk</i>: Input signal is synchronized by system clock. - <i>Digital filter result</i>: Input signal is filtered. - <i>Latched result</i>: Input signal is filtered and is latched when it becomes active.
Low Filter Width (us)	Specify the filter window width for low side comparator, in us. This information is valid only when 'Low Output Sync.' is selected as 'Digital filter result' or 'Latched result'.
Low Threshold (%)	Specify the filter threshold for Low side comparator, in %. This information is valid only when 'Low Output Sync.' is selected as 'Digital filter result' or 'Latched result'.

10.10.2 Comparator DAC

Each comparator DAC has two 12-bit side D/A converters. The outputs of side comparator DAC are linked to the corresponding inverting inputs (Input B) of the corresponding side comparator. Also, a comparator DAC block cannot be used alone, it must be used with a comparator input block and they have the same comparator source

Image:



Attributes:

Parameters	Description
Comparator Source	Comparator source, it can be any of the 8 comparators.
DAC Voltage Reference	Voltage reference of inner DACs, it can be either ' <i>Use VDDA(3.3V)</i> ' or ' <i>Use VDAC</i> '.

DACH Usage	<p>The other signal of the high side comparator:</p> <ul style="list-style-type: none"> - <i>Do not use DACH</i>: The high side inner DAC is not used, the other signal of high side comparator is Input B if the high side comparator is used. - <i>Use constant DAC value</i>: High side inner DAC is used but the DAC output is a constant. The generated program only sets this constant value in the initialization. - <i>Use variable DAC value</i>: High side inner DAC is used and the generated program sets inner DAC value according to its sampling frequency. - <i>Use ramp(Constant)</i>: High side inner DAC is set by ramp generator, the maximum value of ramp generator is a constant. The generated program only sets this constant value in the initialization. - <i>Use ramp(Variable)</i>: High side inner DAC is set by ramp generator, the generated program sets the maximum value of ramp generator according to its sampling frequency.
DACH Range	The input range of high side inner DAC.
Initial DACH Value	Initial value of high side inner DAC. This value is set at the initialization to DACH/Ramp if DACH/Ramp only uses a constant value.
Total Ramp Compensation	Total compensation of the ramp generator in one PWM period. It represents the total decrease of the ramp in one cycle.
Ramp Delay (us)	The delay time in the beginning of PWM period, Ramp decreases when delay time is over, in us.
DACL Usage	<p>Specify the another signal of low side comparator. It can be one of the following:</p> <ul style="list-style-type: none"> - <i>Do not use DACL</i>: The low side inner DAC is not used, the other signal of high side comparator is Input B if the low side comparator is used. - <i>Use constant DAC value</i>: Low side inner DAC is used but the DAC output is a constant. The generated program only sets this constant value in the initialization. - <i>Use variable DAC value</i>: Low side inner DAC is used and the generated program sets inner DAC value according to its sampling frequency.
DACL Range	The input range of low side inner DAC.
Initial DACL Value	Initial value of low side inner DAC. This value is set at the initialization to DACL if DACL only uses a constant value.

Ramp generator only exists in high side inner DAC. When ramp generator is used, a synchronization PWM should be specified in the corresponding Comparator Input block. The input value is applied to the maximum value of ramp generator, ramp generator decreases after the delay time from the beginning of the sync. PWM period is over. When high side comparator's result becomes high, ramp generator restore ramp value to the maximum value and stop decrease, so the active result only remain a short time. It's better to choose 'Latched result' for parameter 'High Output Sync.' in Comparator Input block. The latched result resets at the beginning of the next PWM period.

If the inner DAC is used, the input value is applied to DAC output immediately when there is no sync. PWM specified in Comparator Input block; otherwise the input value is applied to DAC output at the beginning of the sync. PWM's next period.

The voltage reference of inner DAC can be VDDA(3.3V) or VDAC. If VDAC is chose, ADC-B0 is used as VDAC. In Psim schematic, this should be only specified in Voltage Reference block, no need to link a voltage source to ADC-B0. But in target hardware, a voltage reference should be linked to ADC-B0. The output can be calculated as follows:

Use VDDA:

$$DAC_{Output} = DAC_{Input} * 3.3 / DAC_{Range}$$

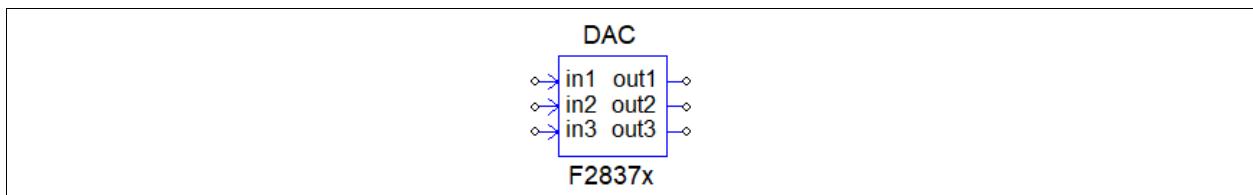
Use VDAC (the voltage reference is specified in Voltage Reference block):

$$DAC_{Output} = DAC_{Input} * VDAC / DAC_{Range}$$

10.11 D/A Converter

F2837x supports three 12-bit buffered D/A converter outputs.

Image:



Attributes:

Parameters	Description
Usage of DAC- <i>x</i>	The voltage reference used by DAC- <i>x</i> , where <i>x</i> is <i>A, B, or C</i> . It can be one of the followings: <ul style="list-style-type: none"> - <i>Do not use DAC-A</i>: DAC-<i>x</i> is not used. - <i>ADC Voltage Reference</i>: DAC-<i>x</i> is used in the system and ADC-<i>x</i> voltage reference is chose by DAC-<i>x</i>. - <i>VDAC Voltage Reference</i>: DAC-<i>x</i> is used in the system and VDAC voltage reference is chose by DAC-<i>x</i>.
DAC- <i>x</i> Input Range	The input range of DAC- <i>x</i> , where <i>x</i> is <i>A, B, or C</i> .
Use PWM Sync. for ADC- <i>x</i>	Specify if DAC- <i>x</i> is synchronized with a PWM block, where <i>x</i> is <i>A, B, or C</i> . The choices are <i>Do not use</i> or one of the PWM blocks in the pull-down list.
DAC- <i>x</i> Initial Value	Initial value of DAC- <i>x</i> , where <i>x</i> is <i>A, B, or C</i> .

If using VDAC as voltage reference, DACs and Comparator DACs can use VDAC as voltage reference; otherwise VDAC can not be selected by DACs and Comparator DACs. Psim does not need ADC-B0 to connect a voltage reference in schematic, but in target system, ADC-B0 must connect a voltage reference.

The voltage reference of DACs can be VDAC or the voltage reference of ADC-A/B. If VDAC is chose, ADC-B0 must be set as VDAC. In Psim schematic, this should be only specified in Voltage Reference block, no need to link a voltage source to ADC-B0. But in target hardware, a voltage reference should be linked to ADC-B0. The output can be calculated as follows:

To use VDAC (the voltage reference is specified in Voltage Reference block):

$$DAC_{Output} = DAC_{Input} * VDAC / DAC_{Range}$$

To use ADC-A voltage reference (for DAC-A and DAC-B only):

$$DAC_{Output} = DAC_{Input} * Vref_ADC-A / DAC_{Range}$$

To use ADC-B voltage reference (for DAC-C only):

$$DAC_{Output} = DAC_{Input} * Vref_ADC-B / DAC_{Range}$$

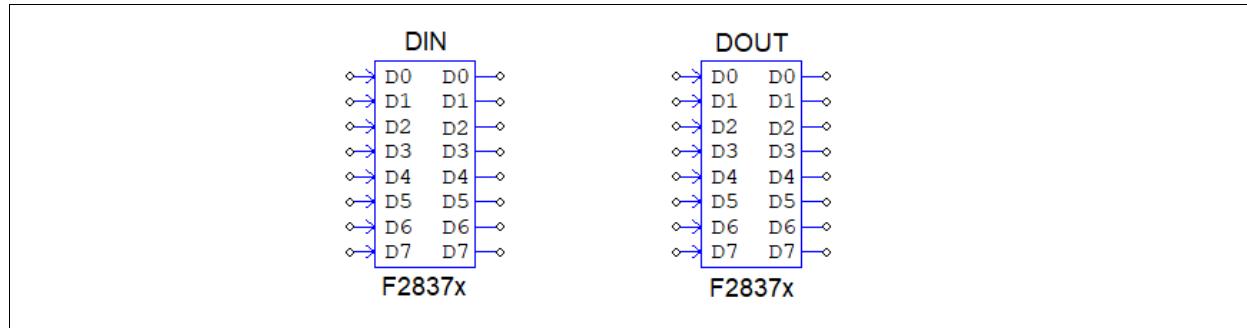
10.12 Digital Input and Digital Output

F2837x support up to 169 individually programmable, multiplexed General-Purpose Input/Output (GPIO) ports with input filtering.

In SimCoder, the digital inputs and outputs are grouped in 8-channel blocks. Multiple 8-channel digital input/output blocks can be used in the same schematic.

All unused inputs should be connected to ground.

Images:



Attributes for Digital Input:

Parameters	Description
Port Position for Input i	The port position of the Input i , where i is from 0 to 7. It can be any one of the GPIO ports.
Invert the Input i	Specify if the signal at Input i is inverted or not.
Use Pull-Up Resistor for Input i	Specify if the Input i has a pull-up resistor at the GPIO port.
Qualification for Input i	There are following qualification options: - Synchronize to CPU clock - 3 sampling period - 6 sampling period - No sync. or qualification

Attributes for Digital Output:

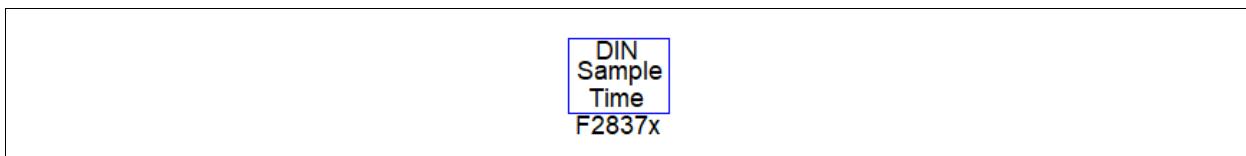
Parameters	Description
Port Position for Output i	The port position of the Output i , where i is from 0 to 7. It can be any one of the GPIO ports.
Port Mode for Output i	Port mode of the output i , where i ranges from 0 to 7. It can be any of the followings: - Normal output - Output with pull-up resistor - Open drain output

Note that each GPIO port can be used for one function only. If a GPIO port is used as a digital input port, it can not be used as a digital output or any other peripheral port. For example, if Port GPIO1 is assigned as digital input and it is also used as PWM1 output, an error will be reported.

10.13 Digital Input Sample Time

This block is used to set the sample time for digital input ports. It is used for code generation only, not in simulation.

Image:



Attributes:

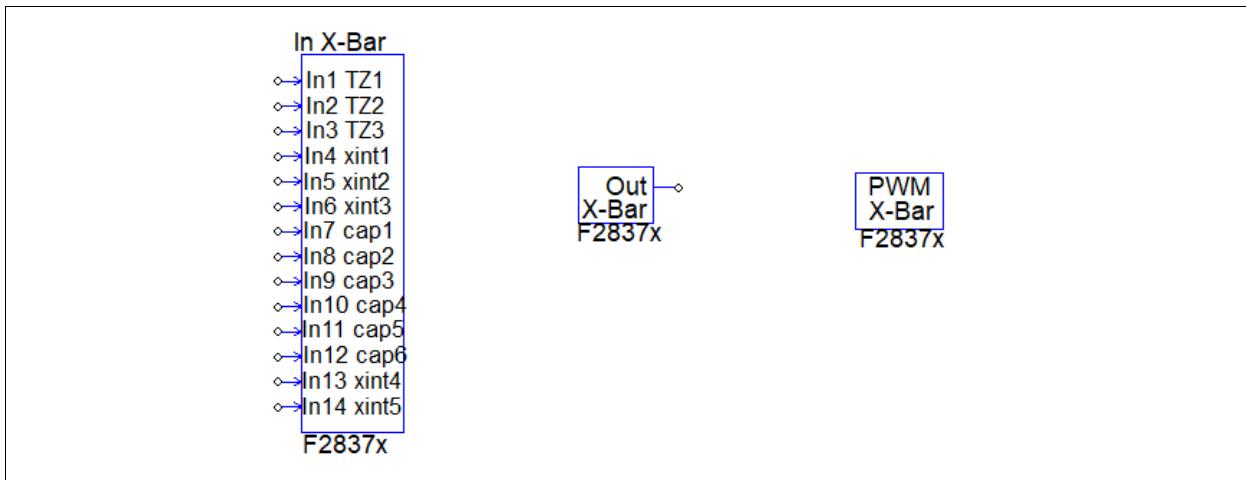
Parameters	Description
Set Gpio $8*i - 8*i+7$ Sample Time	Specify if setting sample time is used for GPIO group i , where ranges from 0 to 21.
Gpio $8*i - 8*i+7$ Sample Time (us)	The sample time for GPIO group i , where i ranges from 0 to 21. The sampling time unit is us.

10.14 Input X-BAR, Output X-BAR, and PWM X-BAR

The crossbars (referred to as X-BAR throughout this document) provide flexibility to connect device inputs, outputs, and internal resources in a variety of configurations. The F2837x device contains three X-BARs. Each of the X-BARs is named according to where they take signals:

- Input X-BAR: brings external signals “in” from a GPIO to the device
- Output X-BAR: takes internal signals “out” of the device to a GPIO.
- PWM X-BAR: takes signals and brings them to the PWM modules.

Images:



10.14.1 Input X-BAR

No more than one Input X-BAR is allowed in each F2837x system.

Attributes:

Parameters	Description
GPIO No. for Input i	Specify which GPIO port is used as input X-Bar signal i

Any GPIO port used as Input X-Bar signal can also be used as any other functionality. For example, GPIO0 is

used as PWM1A, but it can also be used as Input X-Bar signal.

There are 14 input X-Bar signals, They are INPUT1, INPUT2, ..., INPUT14.

- INPUT1, INPUT2 and INPUT3 can be used as trip-zone signals: TZ1, TZ2 and TZ3.
- INPUT4, INPUT5, INPUT6, INPUT13 and INPUT14 can be used as external interrupt sources: XINT1, XINT2, XINT3, XINT4 and XINT5.
- INPUT7, INPUT8, INPUT9, INPUT10, INPUT11 and INPUT12 can be used as capture input signals, the corresponding input capture are Capture1, Capture2, Capture3, Capture4, capture5 and Capture6.
- INPUT1 - INPUT6 can be also used in PWM X-Bar and Output X-Bar by the following names: INPUTXBAR1 - INPUTXBAR6.
- INPUT7 - INPUT12 can be also used in PWM X-Bar and Output X-Bar by the following names: ECAP1OUT - ECAP6OUT.

10.14.2 Output X-BAR

There are 8 Output X-Bar signals, They are OUTPUT1 through OUTPUT8. These signals can be used to take the inner signals out to GPIO ports.

Attributes:

Parameters	Description
Output X-BAR No.	Specify Output X-Bar signal. It can be one of Output1 through Output8. Each output can be linked to one of multiple pre-decided GPIO ports.
Invert the Output	Specify if to invert the output.
Output Type	Specify GPIO output type. It can be one of the followings: - <i>Normal Output</i> : Normal output without pull-up resistor. - <i>Output with pull-up</i> : GPIO port links to an inner pull-up resistor. - <i>Open drain output</i> : GPIO port is an open drain output (SimCoder doesn't support this feature in simulation).
Signal Selection for MUX <i>i</i>	Specify the PWM X-Bar signal, as explained below.

SimCoder only supports the following signals as the source of the output X-Bar. Multiple input sources are allowed. The output is the OR of all selected sources.

MUX No.	Function 0	Function 1	Function 2	Function 3
0	CMPSS1.CTRIPOUTH	CMPSS1.CTRIPOUTH_OR_CTRIPOUTL	N/A	ECAP1OUT
1	CMPSS1.CTRIPOUTL	INPUTXBAR1	N/A	N/A
2	CMPSS2.CTRIPOUTH	CMPSS2.CTRIPOUTH_OR_CTRIPOUTL	N/A	ECAP2OUT
3	CMPSS2.CTRIPOUTL	INPUTXBAR2	N/A	N/A
4	CMPSS3.CTRIPOUTH	CMPSS3.CTRIPOUTH_OR_CTRIPOUTL	N/A	ECAP3OUT
5	CMPSS3.CTRIPOUTL	INPUTXBAR3	N/A	N/A
6	CMPSS4.CTRIPOUTH	CMPSS4.CTRIPOUTH_OR_CTRIPOUTL	N/A	ECAP4OUT
7	CMPSS4.CTRIPOUTL	INPUTXBAR4	N/A	N/A
8	CMPSS5.CTRIPOUTH	CMPSS5.CTRIPOUTH_OR_CTRIPOUTL	N/A	ECAP5OUT
9	CMPSS5.CTRIPOUTL	INPUTXBAR5	N/A	N/A
10	CMPSS6.CTRIPOUTH	CMPSS6.CTRIPOUTH_OR_CTRIPOUTL	N/A	ECAP6OUT
11	CMPSS6.CTRIPOUTL	INPUTXBAR6	N/A	N/A
12	CMPSS7.CTRIPOUTH	CMPSS7.CTRIPOUTH_OR_CTRIPOUTL	N/A	N/A

13	CMPSS7.CTRIPOUTL	N/A	N/A	N/A
14	CMPSS8.CTRIPOUTH	CMPSS8.CTRIPOUTH_OR_CTRIPOUTL	N/A	N/A
15	CMPSS8.CTRIPOUTL	N/A	N/A	N/A

10.14.3 PWM X-BAR

All F2837x's PWM DC trip-zone signals come from the PWM X-Bar,

Attributes:

Parameters	Description
PWM X-BAR No.	Specify PWM X-Bar signal. It can be one of the followings: <i>TRIP4</i> , <i>TRIP5</i> , and <i>TRIP7</i> through <i>TRIP12</i> . There is no <i>TRIP6</i> .
Invert the Output	Specify if to invert the output.
Signal Selection for MUX <i>i</i>	Specify the PWM X-Bar signal, as explained below.

There are 11 PWM X-Bar signals, They are TRIP1 - TRIP5, TRIP7 - TRIP12. There is no TRIP6 signal. These signals are used as DC trip-zone signals for PWM.

TRIP1 - TRIP3 are directly wired to TZ1 - TZ3 in Input X-Bar. The other signals need to set in the PWM X-Bar element. SimCoder only supports the following signals as PWM X-Bar input source. Multiple input sources are allowed. The output is the logic OR of all selected input sources.

MUX No.	Function 0	Function 1	Function 2	Function 3
0	CMPSS1.CTRIPH	CMPSS1.CTRIPH_OR_CTRIPL	N/A	ECAP1OUT
1	CMPSS1.CTRIPL	INPUTXBAR1	N/A	N/A
2	CMPSS2.CTRIPH	CMPSS2.CTRIPH_OR_CTRIPL	N/A	ECAP2OUT
3	CMPSS2.CTRIPL	INPUTXBAR2	N/A	N/A
4	CMPSS3.CTRIPH	CMPSS3.CTRIPH_OR_CTRIPL	N/A	ECAP3OUT
5	CMPSS3.CTRIPL	INPUTXBAR3	N/A	N/A
6	CMPSS4.CTRIPH	CMPSS4.CTRIPH_OR_CTRIPL	N/A	ECAP4OUT
7	CMPSS4.CTRIPL	INPUTXBAR4	N/A	N/A
8	CMPSS5.CTRIPH	CMPSS5.CTRIPH_OR_CTRIPL	N/A	ECAP5OUT
9	CMPSS5.CTRIPL	INPUTXBAR5	N/A	N/A
10	CMPSS6.CTRIPH	CMPSS6.CTRIPH_OR_CTRIPL	N/A	ECAP6OUT
11	CMPSS6.CTRIPL	INPUTXBAR6	N/A	N/A
12	CMPSS7.CTRIPH	CMPSS7.CTRIPH_OR_CTRIPL	N/A	N/A
13	CMPSS7.CTRIPL	N/A	N/A	N/A
14	CMPSS8.CTRIPH	CMPSS8.CTRIPH_OR_CTRIPL	N/A	N/A
15	CMPSS8.CTRIPL	N/A	N/A	N/A

10.15 Encoder, Encoder State, and Encoder Index/Strobe Position

PSIM provides three blocks to execute encoder functions: **Encoder** block, **Encoder State** block, and **Encoder Index/Strobe Position** block.

F2837x supports three **Encoder** modules. Each Encoder module has several GPIO port options. For each GPIO option:

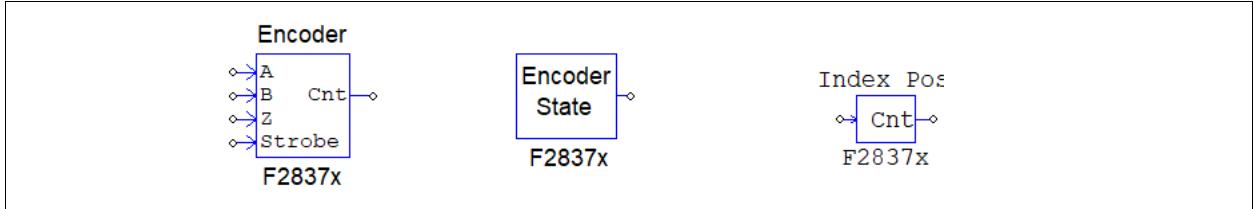
- The 1st GPIO is for the clock input

- The 2nd GPIO is for direction input
- The 3rd GPIO is for Z (or index signal) input
- The 4th GPIO is for strobe signal input

The **Encoder State block** is used to indicate which input signal (either index signal or strobe signal) generates the interrupt. Also, hardware interrupt can be generated by the Z (index) signal and the strobe signal, and the output of the encoder state indicates which signal generates the interrupt. When the output is 0, the index signal generates the interrupt. When the output is 1, the strobe signal generates the interrupt.

The **Encoder Index/Strobe Position block** is used to latch the encoder's initial position. When the input of this block is 0, the encoder counter is set to 0. Encoder will start to act when the input changes to 1. Encoder will latch the counter value when it meet the index/strobe event.

Images:



Attributes for Encoder:

Parameters	Description
Encoder Source	The source of the encoder. It can be any of the 3 encoders and with any of the GPIO port options.
Use Z Signal	Define if the encoder uses the Z (or index) signal.
Use Strobe Signal	Define if the encoder uses the strobe signal.
Counting Direction	The counting direction <ul style="list-style-type: none"> - Forward: the encoder counts up. - Reverse: the encoder counts down.
Z Signal Polarity	Define the trigger polarity of Z signal. <ul style="list-style-type: none"> - Active High - Active Low.
Strobe Signal Polarity	Define the trigger polarity of strobe signal. <ul style="list-style-type: none"> - Active High - Active Low.
Encoder Resolution	The resolution of the external encoder hardware. If it is 0, the encoder counter will keep on counting and will not reset. If for example, the resolution is set to 4096, the counter will be reset to 0 after it reaches 4095.

Attributes for Encoder State:

Parameters	Description
Encoder Source	The source of the encoder. It can be any of the 3 encoders and with any of the GPIO port options.

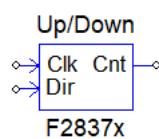
Attributes for Encoder Index/Strobe Position:

Parameters	Description
Encoder Source	The source of the encoder. It can be any of the 3 encoders and with any of the GPIO port options.
Latch Position	Specify the kept counter type, choose from the followings: - <i>IndexPos</i> , if the setting "Use Z Signal" is not "No" in Encoder - <i>StrobePos</i> , if the setting "Use Strobe Signal" is not "No" in Encoder
Type of Position	This can be chosen from the followings: - <i>The first latched position</i> , or - <i>The current latched position</i>

10.16 Up/Down Counter

F2837x supports one up/down counter for each of the three encoder modules.

Image:



Attributes:

Parameters	Description
Counter Source	The source of the counter. It can be any of the 3 encoders and with any of the GPIO port options listed in the pull-down menu.

There are two inputs for Up/Down Counter:

- Input "Clk" refers to the input clock signal.
- Input "Dir" refers defines the counting direction.

When the "Dir" input is 1, the counter counts forward (up), and when the input is 0, the counter counts backward (down).

The output of the up/down counter is the counter value.

Note that the up/down counter shares the same input GPIO ports with the encoder. Therefore, using both encoder and up/down counter on the same GPIO ports is not allowed.

10.17 Capture and Capture State

F2837x contains six enhanced capture module. PSIM use Capture block and Capture State block for capture function setup.

Image:



Attributes for Capture:

Parameters	Description
Capture Source	Source of the capture. It may come from one of the six captures.
Event Filter Prescale	Event filter prescale. The input signal is divided by the selected prescale.
Timer Mode	Capture counter timer mode. It can be one of the followings: - <i>Absolute time</i> : The output is the absolute time in system clock - <i>Time difference (clock)</i> : The output is the difference between the current event and the previous event, in system clock. - <i>Time Difference (us)</i> : The output is the difference between the current event and the previous event, in us.

A capture can generate interrupt, and the interrupt trigger mode is defined in the interrupt block. If there is no the corresponding interrupt block in the schematic, Capture will catch rising edge only.

Capture input comes from Input X-Bar, so if a capture is used in the system, the corresponding input signal of Input X-Bar must be defined too.

Attributes for Capture State:

Parameters	Description
Capture Source	Source of the capture. It has only one source <i>Capture1</i> .

The Capture State block indicates the triggering state of the capture. Its output is either 1 or 0. 1 means the rising edge and 0 means the falling edge.

10.18 Serial Communication Interface (SCI)

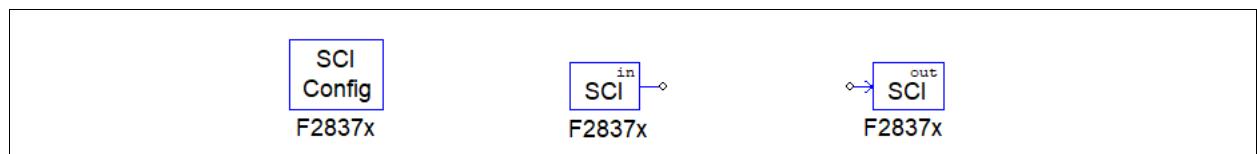
F2837x provides the function for serial communication interface (SCI) functions.

Through SCI, data transferred between the DSP and a computer using an external RS-232 cable. PSIM provides all the necessary functions to transmit and receive data on both the DSP and computer sides, as well as display data on the computer. This provides a very convenient way to monitor, debug, and adjust the DSP code in real time.

For more detailed descriptions on SCI and how to use the monitoring function, please refer to the document "Tutorial - Using SCI for Real-Time Monitoring.pdf".

Three SCI function blocks are provided in SimCoder: *SCI Configuration*, *SCI Input*, and *SCI Output*, as described below.

Images:



10.18.1 SCI Configuration

The SCI Configuration block defines the SCI port, the communication speed, the parity check type, and the data buffer size.

Attributes:

Parameters	Description
Port Selection	F2803x has 4 groups of ports that can be used for SCI functions: Group A, B, D and D. Each group has different port options, as listed in the pull-down menu.
Speed (bps)	SCI communication speed, in bps (bits per second). A list of preset speeds is provided at 200000, 115200, 57600, 38400, 19200, or 9600 bps. Or one can specify any other speed manually.
Parity Check	The parity check setting for error check in communication. It can be either <i>None</i> , <i>Odd</i> , or <i>Even</i> .
Output Buffer Size	Size of the data buffer allocated in DSP for SCI. The buffer is located in the RAM area, and each buffer cell stores one data point which consists of three 16-bit words (that is, 6 bytes, or 48 bits, per data point).

Note that the buffer size should be properly selected. On one hand, a large buffer is preferred in order to collect more data points so that more variables can be monitored over a longer period of time. On the other hand, the internal DSP memory is limited, and the buffer should not be too large to interfere with the normal DSP operation.

10.18.2 SCI Input

The SCI Input block defines a variable in the DSP code that can be changed. The name of the variable will appear in the DSP Oscilloscope (under the **Utilities** menu), and the value can be changed at runtime via SCI.

Attributes:

Parameters	Description
Initial Value	The initial value of the SCI input variable.

In the schematic, the SCI input behaves as a constant. While its value can be changed at runtime when the code is running on the DSP target. The value will be fixed at the initial value in simulation.

10.18.3 SCI Output

The SCI Output block is used to define a variable for display. When a SCI output block is connected to a node, the name of the SCI output block will appear in the DSP Oscilloscope (under the **Utilities** menu), and data of this variable can be transmitted from DSP to the computer via SCI at runtime, and the waveform can be displayed in the DSP Oscilloscope.

The SCI output block provides a convenient way to monitor DSP waveforms.

Attributes:

Parameters	Description
Data Point Step	It defines how frequent data is collected. If the Data Point Step is 1, every data point is collected and transmitted. If the Data Point Step is 10, for example, only one point out of every 10 points is collected and transmitted.

Note that if the Data Point Step is too small, there may be too many data points and it may not be possible to transmit them all. In this case, some data points will be discarded during the data transmission.

Also, the Data Point Step parameter is used only when the DSP Oscilloscope is in the *continuous* mode. When it is in the *snapshot* mode, this parameter is ignored and every point is collected and transmitted.

In simulation, the SCI output behaves as a voltage probe.

10.19 Serial Peripheral Interface (SPI)

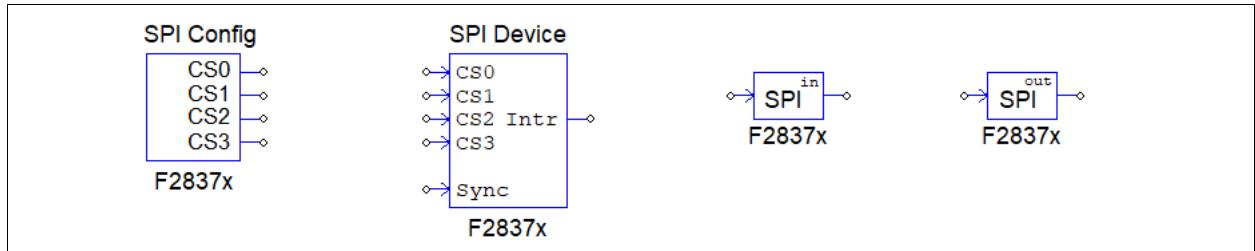
F2837x provides the functions for serial peripheral interface (SPI). The SPI blocks in the TI F2837x Target library provide convenience to implement the function for communication with external SPI devices (such as external A/D and D/A converters).

Writing code manually for SPI devices is often a time-consuming and non-trivial task. With the capability to support SPI, PSIM greatly simplifies and speeds up the coding and hardware implementation process.

For more detailed descriptions on how to use SPI blocks, please refer to the document "*Tutorial - Using SPI for Real-Time Monitoring.pdf*".

Four SPI function blocks are provided in SimCoder: *SPI Configuration*, *SPI Device*, *SPI Input*, and *SPI Output*, as described below.

Images:



10.19.1 SPI Configuration

The SPI Configuration block defines the SPI port, the chip selection pins, and the SPI buffer size. It must be present in a schematic where SPI is used, and this block must be in the main schematic.

Attributes:

Parameters	Description
SPI Port	Define the SPI port from the options. There are three SPI ports: SPIA, SPIB, and SPIC. Each SPI port has a few GPIO pin selections listed in the pull-down menu.
Chip Select Pin0, 1, 2, and 3	The GPIO port of the chip select pin. PSIM supports four GPIO pins for chip select, as defined by Chip Select Pin0 to Pin3. These GPIO ports and the SPI slave transmit-enable pin SPISTE are used to generate the chip select signal.
SPI Buffer Size	The buffer size of the SPI commands. Each memory cell of the buffer saves the index of a SPI command. Normally, one can specify the buffer size as 1 plus the number of SPI commands (i.e. Start Conversion Command, Receiving Data Command, Sending Data Command, and Sync. Command) in all SPI Input/Output elements.

10.19.2 SPI Device

The SPI Device block defines the information of the corresponding SPI hardware device. The number of SPI Device blocks in the schematic must be the same as the number of SPI hardware devices.

Attributes:

Parameters	Description
Chip Select Pins	The state of the chip select pins corresponding to the SPI device. When the chip select pins are at this state, this SPI device is selected.
Communication Speed (MHz)	SPI communication speed, in MHz.

Clock Type	SPI clock type, as determined by the SPI hardware device. It can be one of the following: <ul style="list-style-type: none"> - <i>Rising edge without delay</i>: The clock is normally low, and data is latched at the clock rising edge. - <i>Rising edge with delay</i>: The clock is normally low, and data is latched at the clock rising edge with delay. - <i>Falling edge without delay</i>: The clock is normally high, and data is latched at the clock falling edge. - <i>Falling edge with delay</i>: The clock is normally high, and data is latched at the clock falling edge with delay.
Command Word Length	Word length, or the length of the significant bits, of SPI communication commands. It can be from 1 to 16 bits.
Sync. Active Mode	The triggering mode of the synchronization signal of the SPI device. It can be either <i>Rising edge</i> or <i>Falling edge</i> .
SPI Initial Command	The SPI command that initializes the SPI device.
Hardware Interrupt Mode	Specify the type of the interrupt signal that the SPI device generates. This is valid only when the SPI device's interrupt output node is connected to the input of a digital output element. It can be one of the following: <ul style="list-style-type: none"> - <i>No hardware interrupt</i> - <i>Rising edge</i> - <i>Falling edge</i>
Interrupt Timing	Specify how a SPI device generates interrupt when it completes conversion. It can be one of the following: <ul style="list-style-type: none"> - <i>No interrupt</i>: No interrupt is generated. In this case, DSP sends the command to a SPI input device. This device starts the conversion and returns the result in the same command - <i>Multiple interrupt in series</i>: Multiple interrupts are generated in series after each conversion. This is for a SPI device that has one A/D conversion unit and multiple input channels. In this case, DSP send the first conversion command, and the SPI device starts the conversion. When the conversion is complete, the SPI device will generate an interrupt. In the interrupt service routine, DSP will send a command to fetch the conversion result, and start a new conversion of another channel of the same SPI input device. - <i>One-time interrupt</i>: Only one interrupt is generated at the end of the conversion. This is for a SPI device that can perform multiple channel conversions in one request. In this case, DSP sends the command to the SPI input device, and the SPI device completes the conversion of multiple input channels. When all the conversions are complete, the SPI device will generate an interrupt.
Command Gaps (ns)	The gap between two SPI commands, in nsec.
Conversion Sequence	Define the names of the SPI input elements, separated by comma, that determine the conversion sequence. Note that this parameter is valid only when the SPI device generates multiple interrupts in series.

In a schematic, the chip select pins of all the SPI devices are connected to the chip select pins of the SPI Configuration block, without defining how the chip select logic is implemented. In the actual hardware, however, one would need to implement the corresponding chip select logic accordingly.

A SPI command consists of a series of 16-bit numbers separated by comma. In the 16-bit number, only the lower bits are the significant bits used by the command. For example, if the Command Word Length is 8, Bits 0 to 7 are the command, and Bits 8 to 15 are not used.

A SPI device can be either an input device or an output device. For example, an external A/D converter is an input device. Usually DSP will send one or multiple A/D conversion commands to the device, and then set the

synchronization signal to start the conversion. The synchronization signal is reset at the next command of the same device.

A SPI input device using the synchronization signal usually needs an interrupt pin to trigger DSP to enter the interrupt service routine.

On the other hand, an external D/A converter is an output device. Usually DSP sends one or multiple D/A conversion commands to the device, and then sets the synchronization signal to start the conversion. The synchronization signal is reset at the next command of the same device.

10.19.3 SPI Input

A SPI input device may have multiple input channels. The SPI Input block is used to define the properties of an input channel for SPI communication, and one SPI Input block corresponds to one input channel.

Attributes:

Parameters	Description
Device Name	Name of the SPI input device.
Start Conversion Command	Command to start conversion, in hex numbers, separated by comma (for example, 0x23,0x43,0x00).
Receiving Data Command	Command to receive data, in hex numbers, separated by comma (for example, 0x23,0x43,0x00).
Data Bit Position	Define where the data bits are in the receiving data string. The format is: $\text{ElementName} = \{Xn[\text{MSB..LSB}]\}$ <p>where</p> <ul style="list-style-type: none"> - ElementName is the name of the SPI input device. If it is the current SPI input device, use y instead. - $\{\}$ means that the item in the bracket repeats multiple times. - Xn is the n_{th} word received from the SPI input device, and n start from 0. - MSB..LSB defines the position of the significant bits in the word.
Input Range	Specify the parameter V_{max} that defines the input range. This parameter is valid only when the SPI device is an A/D converter. If the device conversion mode is DC, the input ranges from 0 to V_{max} . If the device conversion mode is AC, the input ranges from $-V_{max}/2$ to $V_{max}/2$.
Scale Factor	Output scale factor K_{scale} . If the scale factor is 0, the SPI device is not an A/D converter, and the result will be exactly the same as what DSP receives from SPI communication. Otherwise, the SPI device is an A/D converter, and the result is scaled based on this factor and the A/D conversion mode.
ADC Mode	The A/D conversion mode of the device. It can be either DC or AC. Note that this parameter is valid only when the device is an A/D converter.
Initial Value	The initial value of the input.

The formula for the *Data Bit Position* defines the data length of a SPI input device. For example, $y=x1[3..0]x2[7..0]$, means that the data length is 12, and the result is the lower 4 bits of the 2nd word and the lower 8 bits of the 3rd word. If the received data string is 0x12,0x78,0xAF, then the result is 0x8AF.

If the scale factor is not 0, the output will be scaled based on the following:

In the DC conversion mode:

- In simulation:
$$\text{Output} = \text{Input} \cdot K_{scale}$$
- In hardware:
$$\text{Output} = \frac{\text{Result} \cdot V_{max} \cdot K_{scale}}{2^{\text{Data_Length}}}$$

In the AC conversion mode:

- In simulation: $Output = Input \cdot K_{scale}$

$$- \text{In hardware: } Output = \frac{(Result - 2^{Data_Length-1}) \cdot V_{max} \cdot K_{scale}}{2^{Data_Length-1}}$$

The parameter *Data_Length* is calculated from the Data Bit Position formula.

10.19.4 SPI Output

A SPI output device may have multiple output channels. The SPI Output block is used to define the properties of an output channel for SPI communication, and one SPI Output block corresponds to one output channel.

Attributes:

Parameters	Description
Device Name	Name of the SPI output device.
Scale Factor	Output scale factor K_{scale} . If the scale factor is 0, the SPI device is not a D/A converter, and the result will be exactly the same as what DSP receives from SPI communication. Otherwise, the SPI device is an D/A converter, and the result is scaled based on this factor and the D/A conversion mode.
Output Range	Specify the parameter V_{max} that defines the output range. This parameter is valid only when the SPI device is an D/A converter. If the device conversion mode is DC, the input ranges from 0 to V_{max} . If the device conversion mode is AC, the input ranges from $-V_{max}/2$ to $V_{max}/2$.
DAC Mode	The D/A conversion mode of the device. It can be either <i>DC</i> or <i>AC</i> . Note that this parameter is valid only when the device is a D/A converter.
Sending Data Command	Command to send the output data, in hex numbers, separated by comma (for example, 0x23,0x43,0x00).
Data Bit Position	Define where the data bits are in the sending data string. The format is: $ElementName = \{Xn[MSB..LSB]\}$ where <ul style="list-style-type: none"> - <i>ElementName</i> is the name of the SPI output device. If it is the current SPI output device, use <i>y</i> instead. - {} means that the item in the bracket repeats multiple times. - <i>Xn</i> is the n_{th} word sent to the SPI output device, and <i>n</i> start from 0. - <i>MSB..LSB</i> defines the position of the significant bits in the word.
Sync. Command	The command to synchronize output channels of the SPI output device, in hex numbers, separated by comma (for example, 0x23,0x43,0x00). This command is used when the SPI output device does not have the synchronization signal

The formula for the *Data Bit Position* defines the data length of a SPI output device. For example, $y=x1[3..0]x2[7..0]$, means that the data length is 12, and the data is the lower 4 bits of the 2nd word and the lower 8 bits of the 3rd word. If the sending data string is 0x12,0x78,0xAF, then the data is 0x8AF.

If the scale factor is not 0, the output will be scaled based on the following:

In the DC conversion mode:

- In simulation: $Output = Input \cdot K_{scale}$

$$- \text{In hardware: } Output = \frac{Result \cdot K_{scale} \cdot 2^{Data_Length}}{V_{max}}$$

In the AC conversion mode:

- In simulation: $Output = Input \cdot K_{scale}$

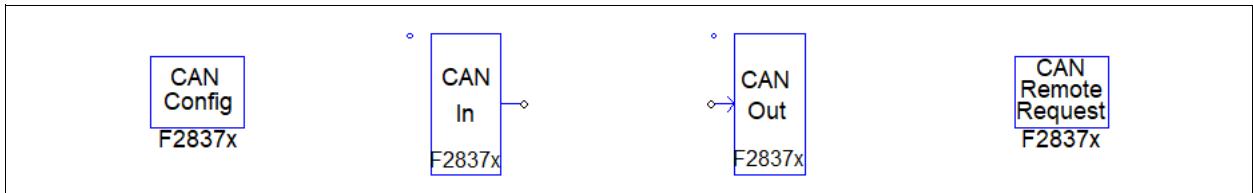
$$\text{- In hardware: } Output = 2^{Data_Length} + \frac{Result \cdot K_{scale} \cdot 2^{Data_Length-1}}{V_{max}}$$

The parameter *Data_Length* is calculated from the Data Bit Position formula.

10.20 Controller Area Network (CAN) Bus

Four function blocks are provided in SimCoder: *CAN Configuration*, *CAN Input*, *CAN Output*, and *CAN Remote Request*, as described below.

Images:



10.20.1 CAN Configuration

The CAN Configuration block defines the CAN bus source, data byte order, and other settings. Attributes:

Parameters	Description
CAN Source	There are 2 sets of CAN devices on F2837x MCU: CAN A and CAN B. Each has multiple GPIO selections in the pull-down list.
CAN Speed	Communication speed, in Hz. It can be set to one of the following preset values: 125kHz, 250kHz, 500kHz, and 1MHz Or you can set the speed manually by typing the text in the parameter field. Note that the speed should not exceed 1MHz.
Data Byte Order	The order of the data bytes. It can be one of the following: - <i>Least significant byte 1st</i> : the least significant byte is placed first - <i>Most significant byte 1st</i> : the most significant byte is placed first.
Number of Input Mailboxes	Number of input mailboxes
Checking Receive Mail Lost	If it is set to <i>Enable</i> , the error report function " <i>_ProcCanAErrReport(nErr)</i> " (for CAN A) will be called if the received mail is lost. This function will return the value of the error status nErr from the CAN register CANGIF0. If it is set to <i>Disable</i> , the error report function will not be called.
Checking Bus Off	If it is set to <i>Enable</i> , the error report function " <i>_ProcCanAErrReport(nErr)</i> " (for CAN A) will be called if the bus is in the off state. This function will return the value of the error status nErr from the CAN register CANGIF0. If it is set to <i>Disable</i> , the error report function will not be called.
Error Check Mode	The error check mode can be either <i>Passive</i> or <i>Active</i> . If it is in the error-passive mode, an interrupt will be generated when the error count reaches 128. If it is in the error-active mode, an interrupt will be generated every time.

The returned status nErr in the function "*_ProcCanAErrReport(nErr)*" (for CAN A) is a 32-bit integer. It obtains its value from the Global Interrupt Flag Register CANGIF0. After returning from the function "*_ProcCanAErrReport(nErr)*", the register CANGIF0 will be cleared.

Also, if you wish to take actions on a specific error, you can add your own code within the "*_ProcCanAErrReport(nErr)*" function.

10.20.2 CAN Input

A CAN Input block receives CAN messages from a CAN bus.

Attributes:

Parameters	Description
Number of Inputs	Number of CAN inputs. It can have up to 8 inputs.
CAN Source	The source of CAN input, either CAN A or CAN B
Use Extension ID	If this is set to <i>Yes</i> , the ID of a message is a 29-bit integer. If this is set to <i>No</i> , the ID of a message is a 11-bit integer.
Message ID	The ID of a message. It is an integer, for example, 0x23.
Local Mask	The mask for the message. It is an integer. If the bits of the message ID matches the bits of the mask, the message will be received. Otherwise, it will be ignored. For example, if the mask is 0x380 and the message ID is 0x389, this message will be received. But if the message ID is 0x480, the message will be ignored.
Overwrite Protection Flag	It can be set to <i>Allow overwrite</i> or <i>Do not allow overwrite</i> . Assume a mailbox is configured to accept a message, and there is a new message coming from the CAN bus, while the old message is still in the mailbox and has not been processed yet. If the flag is set to "Allow overwrite", the new message will be accepted, and the old message will be overwritten. If the flag is set to "Do not allow overwrite", the new message will not be accepted, and the old one will be kept.
Receive Message Rate	The number of messages received by the input block in a specific period of time. For example, there are two input blocks, with the first input block receiving 20 messages per second, and the second input block receiving 30 messages per second. The parameter "Receive Message Rate" will be set to 20 for the first block, and 30 for the second block.
Input i Gain	The gain to the i_{th} input where i can be 1 to 8. The output is the input multiplied by the gain.
Input i Data Start Position	A message can have up to 8 data points. A data point can have 1 bit up to 32 bits. This defines the start position of the current data point in the message.
Input i Data End Position	This defines the end position of the current data point in the message.
Input i Data Type	The data type can be either <i>Float</i> , <i>Integer</i> , or <i>IQ1</i> to <i>IQ30</i> .
Input i Default Data	The initial value of the SCI input variable.

10.20.3 CAN Output

A CAN Output block transmits CAN messages to a CAN bus.

Attributes:

Parameters	Description
Number of Outputs	Number of CAN outputs. It can have up to 8 outputs.
CAN Source	The CAN source can be either CAN A or CAN B.
Use Extension ID	If this is set to <i>Yes</i> , the ID of a message is a 29-bit integer. If this is set to <i>No</i> , the ID of a message is a 11-bit integer.
Message ID	The ID of a message. It is an integer, for example, 0x23.

Trigger Type	<p>The trigger type can be one of the following:</p> <ul style="list-style-type: none"> - <i>No trigger</i>: No triggering - <i>Rising edge</i>: Triggering occurs at the rising edge of the trigger source. - <i>Falling edge</i>: Triggering occurs at the falling edge of the trigger source. - <i>Rising/falling edge</i>: Triggering occurs at both the rising edge and the falling edge of the trigger source. <p>A rising/falling edge is considered to have occurred if the difference between the current value of the trigger source and the value at the last triggering instant is equal to or greater than 1.</p>
Trigger Source	<p>A trigger source can be either a constant or a global variable depending on the Trigger Type.</p> <p>If the Trigger Type is set to "No trigger", the trigger source defines the counter limit. For example, if the trigger source is a constant or a global value, and the value is 5, it means that triggering will occur once out of every 5 times. In another word, the data will be sent out once per every 5 cycles.</p> <p>If the Trigger Type is set to edge trigger, the trigger source can only be a global variable. Triggering will occur when the global variable has the rising or falling edge or both depending on the Trigger Type.</p>
Output <i>i</i> Gain	The gain to the i_{th} output where i can be 1 to 8. The output is the output multiplied by the gain.
Output <i>i</i> Data Start Position	A message can have up to 8 data points. A data point can have 1 bit up to 32 bits. This defines the start position of the current data point in the message.
Output <i>i</i> Data End Position	This defines the end position of the current data point in the message.
Output <i>i</i> Data Type	The data type can be either <i>Float</i> , <i>Integer</i> , or <i>IQ1</i> to <i>IQ30</i> .
Output <i>i</i> Default Data	The initial value of the SCI output variable.

10.20.4 CAN Remote Request

The CAN Remote Request block sends a remote request to CAN bus, and receives the message from the remote frame with the same message ID on CAN bus.

Attributes:

Parameters	Description
CAN Source	The source of CAN input, either CAN A or CAN B
Use Extension ID	If this is set to <i>Yes</i> , the ID of a message is a 29-bit integer. If this is set to <i>No</i> , the ID of a message is a 11-bit integer.
Message ID	The ID of a message. It is an integer, for example, 0x23.
Trigger Type	<p>The trigger type can be one of the following:</p> <ul style="list-style-type: none"> - <i>No trigger</i>: No triggering. - <i>Rising edge</i>: Triggering occurs at the rising edge of the trigger source. - <i>Falling edge</i>: Triggering occurs at the falling edge of the trigger source. - <i>Rising/falling edge</i>: Triggering occurs at both the rising edge and the falling edge of the trigger source. A rising/falling edge is considered to have occurred if the difference between the current value of the trigger source and the value at the last triggering instant is equal to or greater than 1.

Trigger Source	A trigger source can be either a constant or a global variable depending on the Trigger Type. If the Trigger Type is set to "No trigger", the trigger source defines the counter limit. For example, if the trigger source is a constant or a global value, and the value is 5, it means that triggering will occur once out of every 5 times. In another word, the data will be sent out once per every 5 cycles. If the Trigger Type is set to edge trigger, the trigger source can only be a global variable. Triggering will occur when the global variable has the rising or falling edge or both depending on the Trigger Type.
Frequency	The sampling frequency.

10.21 Interrupt Time

The interrupt time block is used to measure the time interval of an interrupt service routine.

Attributes:

Parameters	Description
Time Output Method	Define how interrupt time is measured. It can be one of the following: <ul style="list-style-type: none"> - <i>SCI (time used)</i>: Using SCI. Time used by the interrupt service routine, in DSP clock count, is measured and is sent out via SCI output. - <i>SCI (time remaining)</i>: Using SCI. Time remaining in the interrupt service routine, in DSP clock count, is measured and is send out via SCI output. The time remaining is defined as the time from the end of the current interrupt to the beginning of the next interrupt. - <i>GPIO0 to GPIO44, or AIO2 to AIO14</i>: Using a GPIO port. A pulse is generated at the specified GPIO port. The pulse is set to high when entering the interrupt, and set to low when exiting the interrupt. An oscilloscope can be used to measure the width of the pulse.
Sampling Frequency	Sampling frequency of the interrupt service routine, in Hz.

When SCI is used, the value is the count of the DSP clock. For example, if the value is 7500, for a 90-MHz DSP clock, the interrupt time will be:

$$7500 / 90M = 83.33\text{us}$$

10.22 Project Settings and Memory Allocation

When generating the code for the TI F2837x Hardware Target, SimCoder also creates the complete project files for the TI Code Composer Studio (CCS) development environment, so that the code can be compiled, linked, and uploaded to the DSP.

At the present, CCS version 3.3 is supported. Assuming that the PSIM schematic file is "test.sch", after the code generation, a sub-folder called "test (C code)" will be generated in the directory of the schematic file, and sub-folder will contain the following files:

- test.c Generated C code
- PS_bios.h: Header file for the SimCoder F2837x library
- passwords.asm: File for specifying the DSP code password
- test.pjt: Project file for Code Composer Studio
- F2837x_Headers_nonBIOS.cmd: Peripheral register linker command file
- F2837x_FLASH_Lnk.cmd: Flash memory linker command file
- F2837x_FLASH_RAM_Lnk.cmd: Flash RAM memory linker command file
- F2837x_RAM_Lnk.cmd: RAM memory linker command file

Note: The names of the link command files are assigned with the target hardware if it is not F2837x. For

example, if the target hardware is F28069, the file names will be *F28069 FLASH Lnk.cmd*, *F28069 FLASH RAM Lnk.cmd*, and *F28069RAM Lnk.cmd* accordingly.

These library files will be copied automatically to the project folder when the code is generated.

Each time code generation is performed, the .c file and .pj file (in this example, "test.c" and "test.pjt") will be created. If you have made changes manually to these two files, be sure to copy the changed files to a different location. Otherwise the changes will be overwritten when code generation is performed next time.

Project Setting:

In the Code Composer Studio project file, the following settings are provided:

- *RAM Debug*: To compile the code in the debug mode and run it in the RAM memory
- *RAM Release*: To compile the code in the release mode and run it in the RAM memory
- *Flash Release*: To compile the code in the release mode and run it in the flash memory
- *Flash RAM Release*: To compile the code in the release mode and run it in the RAM memory

When RAM Debug or RAM Release is selected, CCS uses the linker command file *F2837x_RAM_Lnk.cmd* to allocate the program and data space.

When Flash Release is selected, CCS uses the linker command file *F2837x_FLASH_Lnk.cmd* to allocate the program and data space.

When Flash RAM Release is selected, CCS uses the linker command file *F2837x_FLASH_RAM_Lnk.cmd* to allocate the program and data space. The memory allocation is the same as in RAM Release.

The code compiled in the release mode is faster than the code in the debug mode. Also, the code in RAM Release or Flash RAM Release is the fastest. The code in RAM Debug is slower, and the code in Flash Release is the slowest. In a development, normally one would start with RAM Debug for easy debugging. Then switch to RAM Release and consequently to Flash Release or Flash RAM Release.

F28004x Hardware Target

11.1 Overview

With the F28004x Hardware Target, SimCoder can generate code that is ready to run on any hardware boards based on Texas Instruments' F28004x 32-bit floating-point microcontroller unit (MCU).

This chapter describes the definitions and usage of the elements in the F28004x Hardware Target library.

The F28004x Hardware Target library includes the following function blocks:

- PWM generators: 3-phase, 2-phase, 1-phase, and APWM
- Variable frequency PWM
- Start/Stop functions for PWM generators
- PWM trip-zone and trip-zone state
- A/D converter
- ADC voltage reference
- PGA and PGA gain
- Comparator input, output, and DAC
- DAC converter
- Digital input, output, and sample time
- Input X-BAR, output X-BAR, and PWM X-BAR
- Encoder, encoder state, and encoder index/strobe position
- Up/Down counter
- Capture and capture state
- SCI configuration, Input, and output
- SPI configuration, device, input, and output
- CAN configuration, input, output, and remote request
- Interrupt Time
- DSP clock
- Hardware configuration

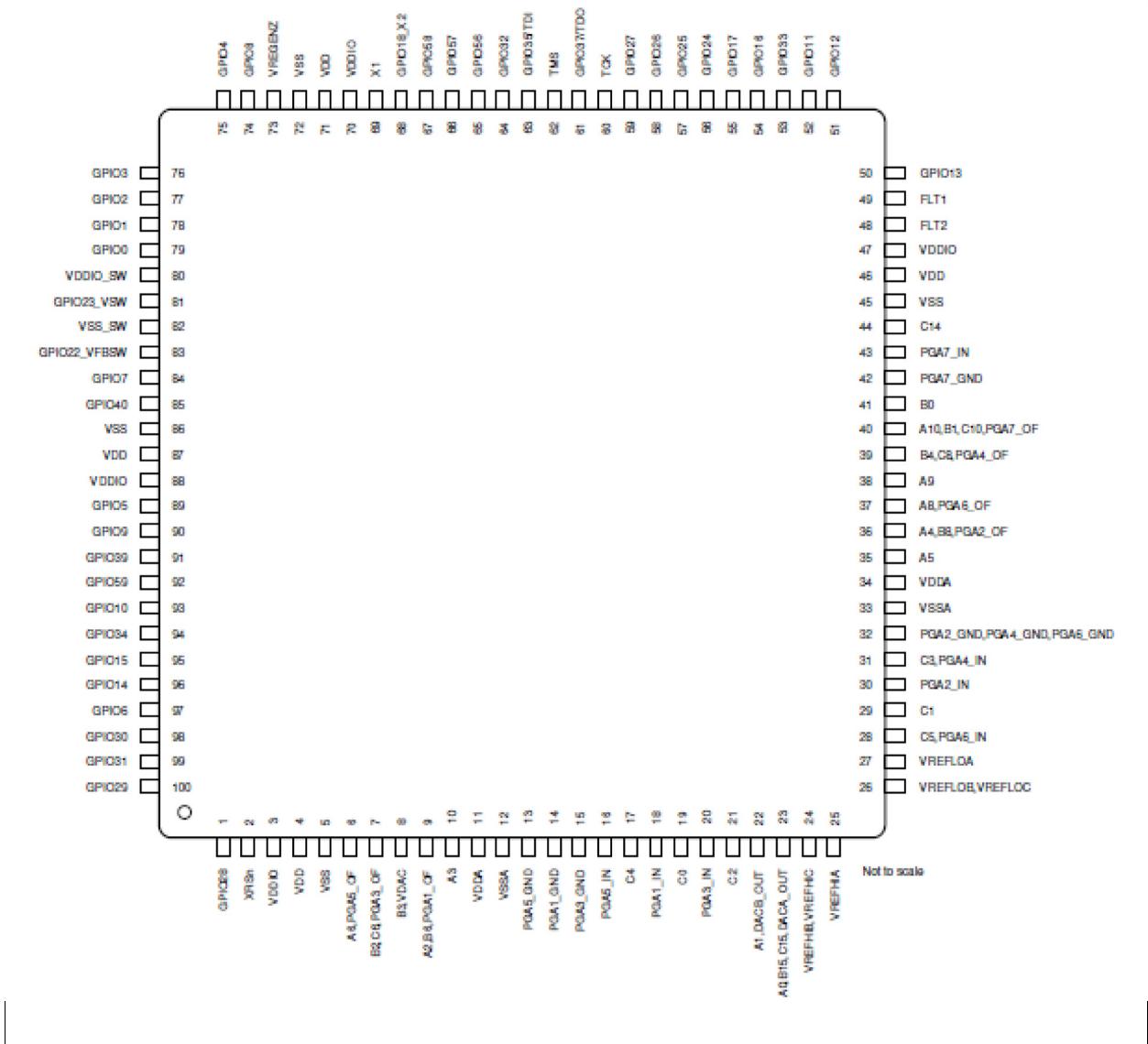
When generating the code for a system that has multiple sampling rates, SimCoder will use the interrupts of the PWM generators for the PWM sampling rates. For other sampling rates in the control system, it will use the Timer 1 interrupt first, and then Timer 2 interrupt if needed. If there are more than three sampling rates in the control system, the corresponding interrupt routines will be handled in the main program by software.

In TI F28004x, PWM generators can generate hardware interrupt. SimCoder will search and group all the elements that are connected to the PWM generator and have the same sampling rate as the PWM generator. These elements will be automatically placed and implemented in an interrupt service routine in the generated code.

In addition, digital input, encoder, capture, and trip-zone can also generate hardware interrupt. Each hardware interrupt must be associated with an interrupt block (described in Section 4.5 of this Manual), and each interrupt block must be associated with an interrupt service routine (a subcircuit that represents the interrupt service routine). For example, if a PWM generator and a digital input both generate interrupt, there should be one interrupt block and one interrupt service routine for each of them.

The F28004x Hardware Target will work with all F28004x packages. The figure below shows the F28004x 100-pin PZ package port assignment. Only the GPIO functions are shown on GPIO pins. The other port functions will be explained in next section about hardware configuration.

F28004x 100-Pin PZP Port Assignment



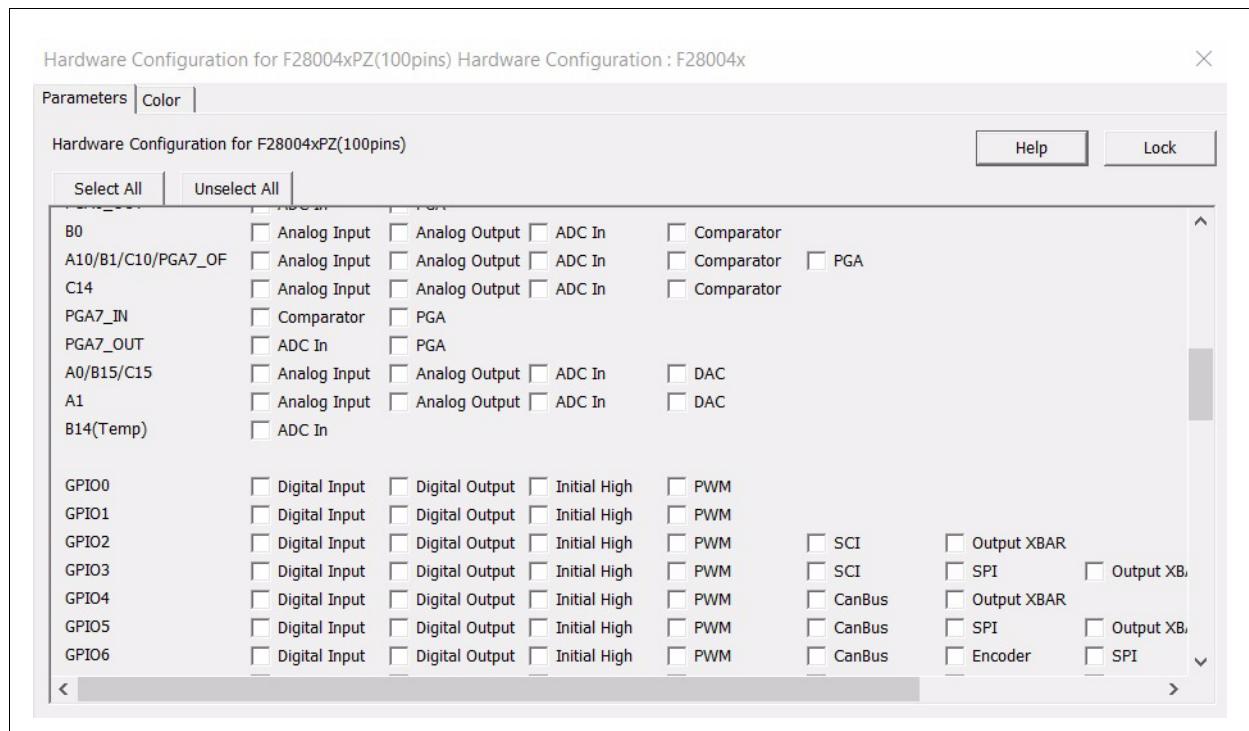
11.2 Hardware Configuration

F28004x provides analog channels and individually programmable multiplexed GPIO ports. Many of the GPIO ports can perform one of several functions. For example, port GPIO0 can be used either as a digital input, a digital output, or a PWM output. Some of the A/D channels can also be defined as a comparator or a PGA input. Therefore, one must correctly assign the functions to the GPIO ports according to the PSIM circuit schematic.

Image:



The dialog window of the block is shown below:



The Hardware Configuration block is for user to specify the I/O ports usage of the F28004x hardware. Every port in use must be assigned correctly. The ports not in use can be left unchecked.

To make changes, click on the button **Unlock**. After changes are made, click on **Lock** to lock the configuration.

Analog Subsystem:

In the analog subsystem, each port has several functionalists, and each functionality is independent of others. One may use any or all functions in one system.

For example, Pin "A2/B6/PGA1_OF" can be used accepting inputs from: ADC-A2, AC-B6, and PGA1_OF. And it can be used as AIO224 input/output.

If multiple functionality boxes are checked, in corresponding ports/pins must be connected in the circuit schematic.

GPIO Ports:

All GPIO ports are set to low at the start up by default. If any GPIO needs to start high, the box "Initial High" must be checked.

For each GPIO port, a check box is provided for each of its available function. Multiple boxes can be checked if

the functions do not conflict with each other.

For example, GPIO0 has 3 functionality check boxes ("Initial High" is not a GPIO functionality): "**Digital Input**", "**Digital Output**" and "**PWM**". One can use GPIO0 as either a "Digital Output" port or a "PWM" output but not both in the same circuit. However, If in a circuit schematic, the PWM1A is connected to GPIO0 in a "Digital Input" block, this port can be used as a "Digital Input" and a "PWM" output together. In such a case, if PWM1A is not connected to GPIO0 in a Digital Input block in the schematic, SimCoder will report an error message.

Similarly, GPIO also be used as a a "Digital Input" and "Digital Output" if a connection is made between the input and the output blocks.

11.3 DSP Clock

The DSP Configuration block defines the DSP clock source and frequency, as well as the DSP speed.

Image:



Attributes:

Parameters	Description	Default Value
DSP Clock Source	There are three ways of providing system clock to F28004x: - Internal oscillator 1 - Internal oscillator 2 - External oscillator	External oscillator
External Clock (MHz)	Frequency of the external clock on the DSP board, in MHz. The frequency must be an integer, and the maximum frequency allowed is 20 MHz. This parameter is ignored if the DSP clock source is selected to be internal oscillator 1 or 2.	20
DSP Speed (MHz)	DSP Speed, in MHz. The speed must be an integer, and must be an integer multiple of the external clock frequency. The maximum DSP speed allowed is 100MHz	100

11.4 PWM Generators

F28004x contains 8 sets of PWM modules for 100pin package. Each set of PWM module has two output ports.

In SimCoder, the PWM's can be used in the following ways:

- **1-phase PWM generators:** with two outputs complementary to each other.
- **1-phase PWM generators with phase shift:** with two outputs complementary to each other.
- **2-phase PWM generators:** with the two outputs of each PWM generator in special operation mode, maybe not in a complementary way.
- **3-phase PWM generators:** consisting three 1-phase PWM generators of consecutive order, such as PWM 123 (consisting of PWM 1, 2, and 3) and PWM 456 (consisting of PWM 4, 5, and 6);

These PWM generators can trigger the A/D converter, and use trip-zone signals.

Beside the PWM generators described above, there are also 7 single output APWM generators that use the same resources as the captures. These APWM generators have restricted functionality comparing to the PWM generators. They can not trigger the A/D converter and can not use trip-zone signals. Also, because of the common resources, when a particular port is used for the capture, it can not be used for the PWM generator.

Note that all the PWM generators in SimCoder include one switching period delay internally. That is, the input value of a PWM generator is delayed by one cycle before it is used to update the PWM output. This delay is

needed to simulate the delay inherent in the DSP hardware implementation.

PWM generators have a parameter called "PWM Freq. Scaling Factor". It can be set to 1 to 15.

PWM generators can generate an interrupt in two ways:

- Periodical interrupt. The interrupt frequency is the same as the sampling frequency. It can be generated in the following ways:
 - If PWM is selected as the interrupt source, interrupt will be generated by PWM itself at the start of the PWM carrier wave.
 - If the A/D converter is selected as the interrupt source, PWM will trigger the A/D converter to start the conversion. After the A/D conversion is complete, interrupt will be generated.
- Trip-Zone interrupt. 3 tripzone singles (TZ1, TZ2 and TZ3) from input X-bar and 4 DC trip events (or their combination) can be used as interrupt source. If the trip-zone signal of the PWM is active, a trip-zone interrupt will be generated. Before entering the trip-zone interrupt, all PWM outputs will be set according to the Trip Action definition. DC trip events are set at the corresponding PWM trip-zone element.

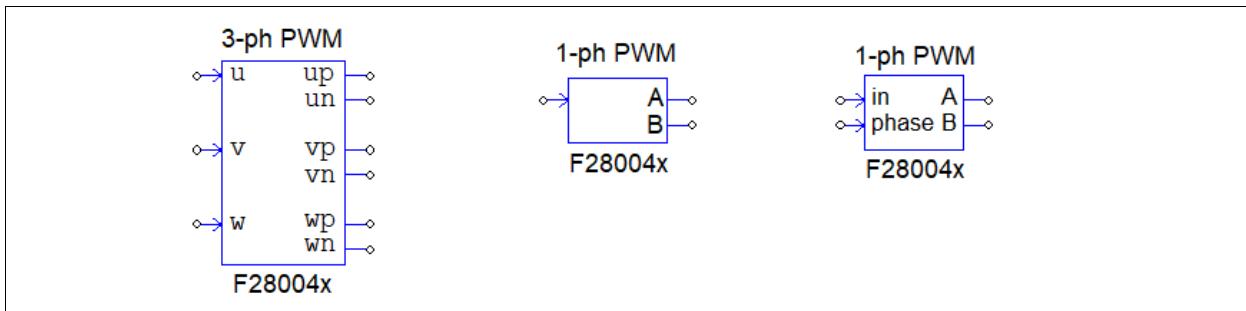
11.4.1 3-Phase PWM, 1-phase PWM, and 1-phase PWM (phase shift)

The 3-phase PWM generator consists of three 1-phase PWM blocks of consecutive order. Therefore, the attributes for **3-phase PWM** block, **1-Phase PWM** block and **1-phase PWM (phase shift)** block are mostly the same.

The difference between the **1-Phase PWM** and **1-phase-PWM (phase shift)** blocks is how the phase shift is defined. The 1-phase PWM block defines the phase shift through a parameter which is a constant, while the 1-Phase PWM (phase shift) block reads the phase shift from an external input (labeled as "phase" in the image). The phase shift is in degree.

In the 3-phase PWM generator image, "u", "v", and "w" refer to the three phases (alternatively they are called phase "a", "b", and "c"). The letter "p" refers to the positive output, and "n" refers to the negative output. For example, for 3-phase PWM 123, "up" is PWM1A, and "un" is PWM1B.

Image:



Attributes:

Parameters	Description
PWM Source	Source of the PWM generator. Each 1-phase PWM block has two outputs (PWM A and PWM B) assigned to two GPIO pins. Each 3-phase PWM module consists of three 1-phase PWM blocks. For example, 3-phase <i>PWM 123 (GPIO0-5)</i> consists of PWM1, PWM2, and PWM3. Therefore, each 3-phase PWM uses 6 GPIO pins.
Output Mode (Not used for 3-phase)	Output mode of the PWM generator. It can be one of the following: <ul style="list-style-type: none"> - <i>Use PWM A&B</i>: Both PWM outputs A and B are used., - <i>Use PWM A</i>: Only PWM output A is used. - <i>Use PWM B</i>: Only PWM output B is used. - <i>No PWM output</i>: No PWM output is used (As a timer).

Dead Band Type	The type of PWM dead band: <ul style="list-style-type: none"> - Active high complementary - Active low complementary - Active high - Active low.
Rising Edge Dead Time (us)	The dead time for the rising edge of the PWM generator, in us.
Falling Edge Dead Time (us)	The dead time for the falling edge of the PWM generator, in us.
Sampling Frequency	Sampling frequency of the PWM generator, in Hz. The calculation is done and the PWM signal duty cycle is updated based on this frequency.
PWM Freq. Scaling Factor	The ratio between the PWM switching frequency and the sampling frequency. It can be from 1 to 15. For example, if the sampling frequency is 50 kHz and the scaling factor is 3, the PWM switching frequency will be 150 kHz. That is switches will operate at 150 kHz. But gating signals will be updated at 50 kHz, or once per 3 switching cycles.
Carrier Wave Type	Carrier wave type and the initial PWM output state. It can be one of the following: <ul style="list-style-type: none"> - <i>Triangular (start low)</i>: Triangular wave, and the initial PWM output state is low. - <i>Triangular (start high)</i>: Triangular wave, and the initial output state is high. - <i>Sawtooth (start low)</i>: Sawtooth wave, and the initial output state is low. - <i>Sawtooth (start high)</i>: Sawtooth wave, with the initial output state is high.
Trigger ADC	Setting whether for the PWM generator to trigger the A/D converter. It can be one of the followings: <ul style="list-style-type: none"> - <i>Do not trigger ADC</i>: PWM does not trigger the A/D converter. - <i>Trigger ADC</i>: PWM triggers the A/D converter channel (channels) connected to this PWM module.
ADC Trigger Position	A/D trigger position ranges between 0 and 1. When it is 0, the A/D converter is triggered at the beginning of the PWM cycle, and when it is 0.5, the A/D converter is triggered at the 180° position of the PWM cycle.
Use Trip-Zone 1, 2, 3	Define whether the PWM generator uses the trip-zone signal or not. It can be one of the followings: <ul style="list-style-type: none"> - <i>Do not use</i>: Disable the i_{th} trip-zone signal. - <i>One shot</i>: The PWM generator uses the trip-zone signal in the one-shot mode. Once triggered, the PWM must be started manually. - <i>Cycle by cycle</i>: The PWM generator uses the trip-zone signal in the cycle-by-cycle basis. The trip-zone signal is effective within the current cycle, and PWM will automatically re-start in the next cycle.
Select Trip Events	Specify the combination of DC trip events applied to the current PWM. It can be one or any combination of DCAEVT1, DCAEVT2, DCBEVT1, and DCBEVT2.
Trip Action	Define how the PWM generator responds to the trip action. It can be one of the followings: <ul style="list-style-type: none"> - <i>High impedance</i>: PWM outputs in high impedance - <i>PWM A high & B low</i>: Set PWM A high and B low. - <i>PWM A low & B high</i>: Set PWM A low and B high. - <i>PWM A both high</i>: Set both PWM A and B high (not used for 3-phase). - <i>PWM A both low</i>: Set both PWM A and B low (not used for 3-phase). - <i>No action</i>: No action taken.
Peak-to-Peak Value	Carrier wave peak-to-peak value V_{pp} , in V.

Offset Value	Carrier wave DC offset value V_{offset} , in V.
Initial Input Value (u, v, w for 3-phase)	Initial value of the PWM block. For three initial input values are set for 3-phase PWM block's three inputs u , v , and w .
Phase Shift (for 1-phase only)	Phase shift of the output with respect to the reference PWM generator output, in deg. For example, when the phase shift is -30, the output will be shifted to the right (lagging) by 30 deg. with respect the reference PWM output.
Use HRPWM (Not used for 3-phase)	Define the high-resolution PWM. It can be one of the followings: <ul style="list-style-type: none"> - <i>Do not use HRPWM</i>: Do not use high-resolution PWM - <i>Use HRPWM without calibration</i>: Use high-resolution PWM without calibration - <i>Use HRPWM with calibration</i>: Use high-resolution PWM with calibration
Start PWM at Beginning	When it is set to <i>Start</i> , PWM will start right from the beginning. If it is set to <i>Do not start</i> , one needs to start PWM using the Start PWM block.
Simulation Output Mode	The simulation output mode can be set to either of the followings: <ul style="list-style-type: none"> - <i>Switching mode</i>: the outputs of the PWM block are PWM signals. - <i>Average mode</i>: the outputs of the PWM block are average mode signals. <p>In the average mode, if the carrier wave is from negative to positive, and the absolute values of the negative peak and the positive peak are equal (for example, the carrier wave is from -1 to +1, or from -5 to +5), the modulation is considered as an ac signal modulation. Otherwise, the modulation is considered as a dc signal modulation. For example, modulation in a 3-phase or single-phase inverter is an ac modulation, and modulation in a buck converter is a dc modulation.</p> <p>In the ac signal modulation, if the input u of the PWM block is V_u, the output up and un in average mode will be:</p> $V_{up} = V_u / (V_{pp} + V_{offset})$ $V_{un} = -V_{up}$ <p>In this case, V_u is between $-(V_{pp} + V_{offset})$ and $(V_{pp} + V_{offset})$, and V_{up} is between -1 to +1.</p> <p>In the dc signal modulation, the output up and un in average mode will be:</p> $V_{up} = (V_u - V_{offset}) / V_{pp}$ $V_{un} = 1 - V_{up}$ <p>In this case, V_u is between V_{offset} and $V_{pp} + V_{offset}$, and V_{up} is between 0 to +1.</p> <p>When it is set to the average mode, the PWM block outputs can be connected to a converter/inverter in the average mode model.</p>

Phase Shift

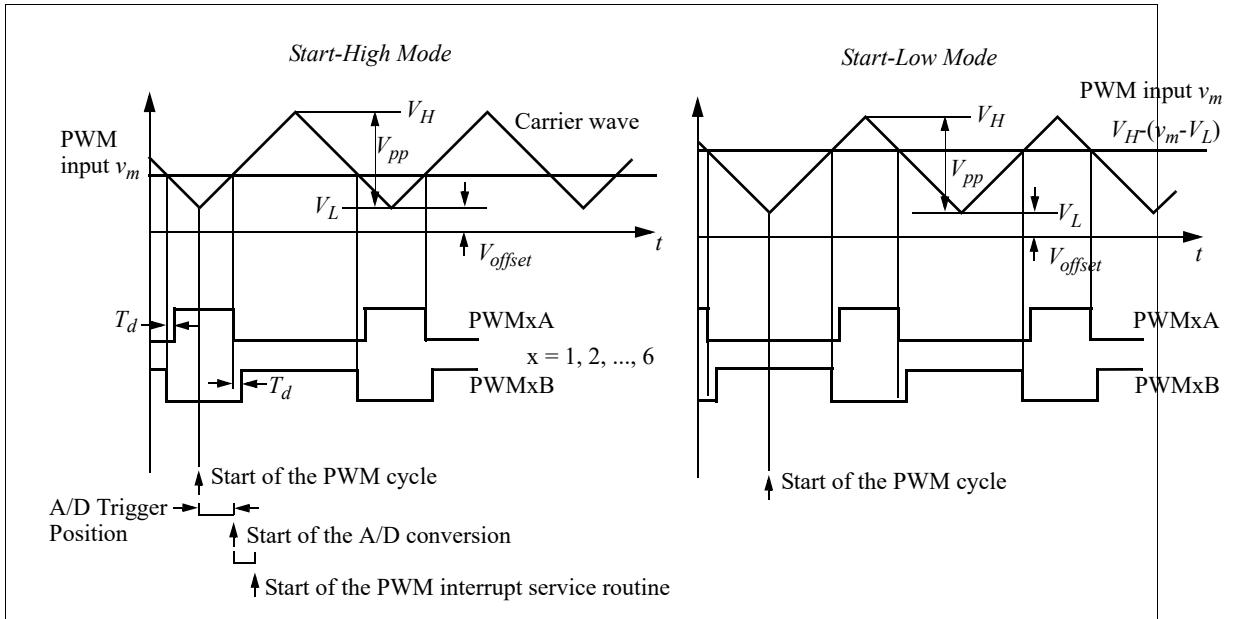
A 1-phase PWM generator can generate PWM signal that is phase shifted with respect to another PWM signal. The way how PWM blocks are defined for phase shift is explained in Section 8.4.5.

The phase shift value is in degrees. When the value is -30°, the output will be shifted to the right (lagging) by 30° of the switching cycle with respect to the reference PWM generator output. This is equivalent to shifting the PWM carrier wave to the right by 30°. When the phase value is 30°, the output will be shifted to the left (leading) by 30°.

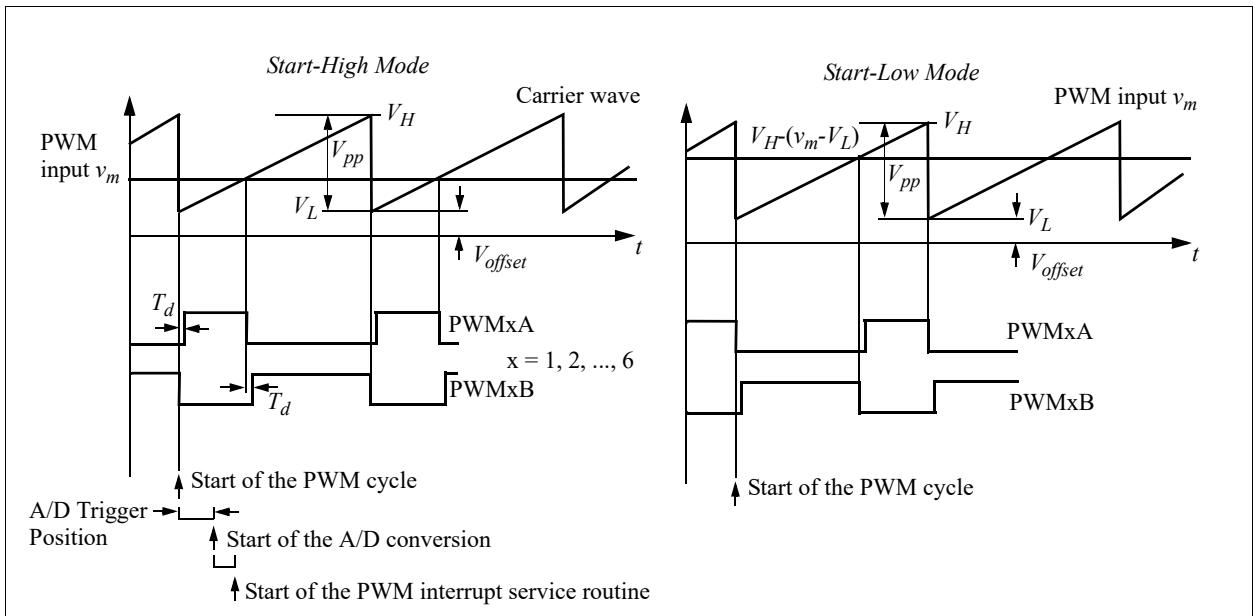
Carrier Wave

There are two types of carrier waveforms: triangular wave (with equal rising and falling slope intervals) and sawtooth wave. In addition, there are two operation modes: start-low and start-high modes, as explained below.

The input and output waveforms of a PWM generator with the triangular carrier wave are shown below:



The input and output waveforms of a PWM generator with the sawtooth carrier wave are shown below:



The figures above show how the dead time is defined, and the time sequence when the PWM generator triggers the A/D converter. If triggering the A/D converter is selected, from the start of the PWM cycle, after a certain delay defined by the A/D trigger position, the A/D conversion will start. After the A/D conversion is completed, the PWM interrupt service routine will start.

If the PWM generator does not trigger the A/D converter, the PWM interrupt service routine will start at the beginning of the PWM cycle.

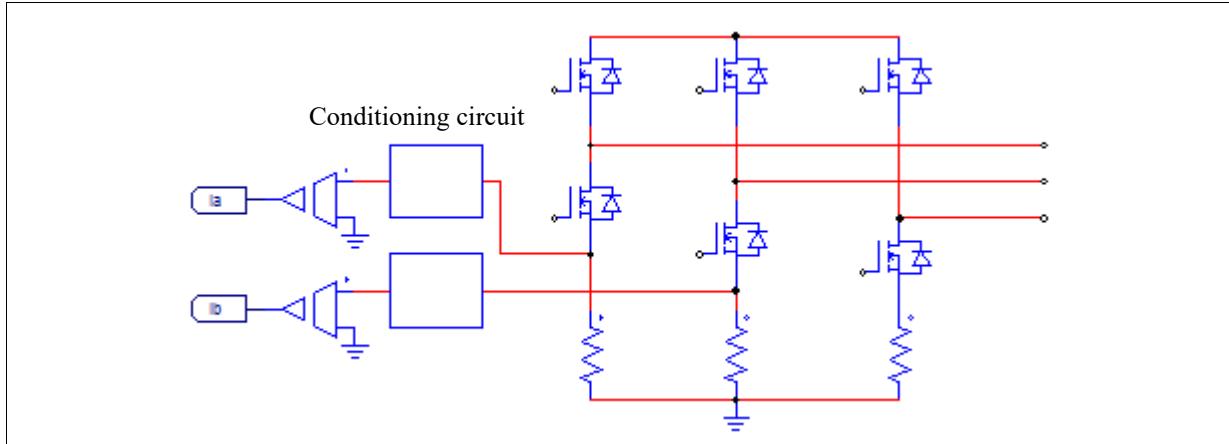
The figures above also show how the start-high and start-low modes work. Assume that the PWM input is v_m , and the lowest value of the carrier wave is V_L and the highest value is V_H . In the start-high mode, the PWM positive output PWMA is high at the beginning of the switching cycle, and it remains high as long as the input v_m is greater than the carrier wave. For example, for a carrier wave from 0 to 1, $V_L=0$, and $V_H=1$. If $v_m=0.2$, the PWM output PWMA will remain high as long as the carrier is less than 0.2.

On the other hand, in the start-low mode, the PWM positive output PWMA is low at the beginning of the switching cycle, and it is high when the carrier wave is greater than the value $V_H(v_m - V_L)$. For example, for a

carrier wave from 0 to 1, $V_L=0$, and $V_H=1$. If $v_m=0.2$, the PWM output PWMA will be high as long as the carrier is greater than 0.8.

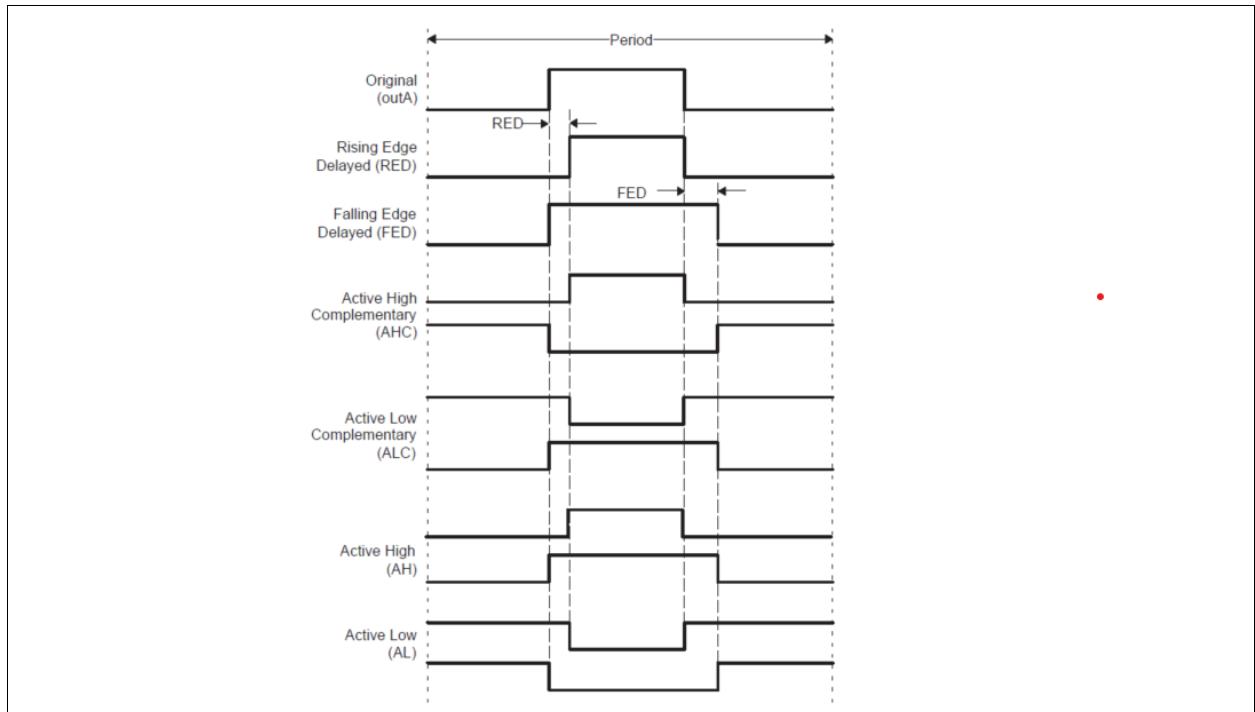
The carrier start mode depends on how switch currents are measured. In a 3-phase inverter, for example, if top switch currents are measured, the start-high mode should be selected. This ensures that at the beginning of the cycle, the top switch gating signal is high and the current is conducting. On the other hand, if bottom switch currents are measured, the start-low mode should be selected. This ensures that at the beginning of the cycle, the top switch gating signal is low, and the bottom switch gating signal is high and the current is conducting.

For example, in the circuit below, the bottom switch currents of Phase A and B are measured. In this case, the carrier start-low mode should be selected.



Note: In the start-low mode, the PWM input v_m is converted to $V_H(v_m-V_L)$ internally before it is compared with the carrier wave to generate the PWM signal. With the conversion, both the start-low and start-high modes will have the same duty cycle expression. For example, for a sawtooth wave with $V_L=0$ and $V_H=1$, or for a triangular wave with $V_L= -V_H$, the duty cycle D of the PWMA output in both the start-low and start-high modes is: $D = v_m/V_H$.

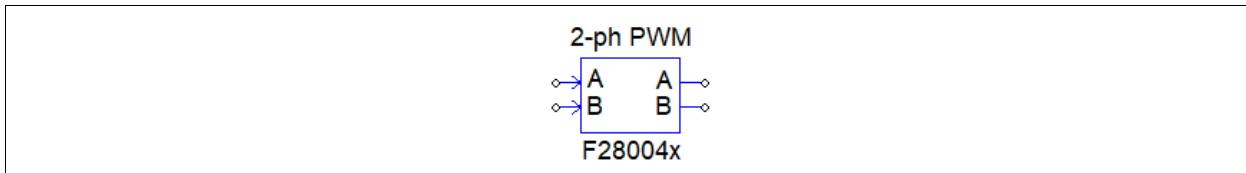
The PWM dead band acts as below.



11.4.2 2-Phase PWM Generator

A 2-phase PWM generator has two inputs and two outputs, and operates in one of the six pre-defined operation modes.

Image:



Attributes:

Parameters	Description
PWM Source	Source of the PWM generator. It can be PWM 1 to PWM 8.
Sampling Frequency	Sampling frequency of the PWM generator, in Hz. The calculation is done and the PWM signal duty cycle is updated based on this frequency.
PWM Freq. Scaling Factor	The ratio between the PWM switching frequency and the sampling frequency. It can be from 1 to 15. For example, if the sampling frequency is 50 kHz and the scaling factor is 3, the PWM switching frequency will be 150 kHz. That is switches will operate at 150 kHz. But gating signals will be updated at 50 kHz, or once per 3 switching cycles.
Carrier Wave Type	There are 2 kind of carrier wave types: <i>Sawtooth</i> and <i>Triangular</i>
PWMxA Output Mode	PWMxA and PWMxB output mode can be one of the followings:
PWMxB Output Mode	<ul style="list-style-type: none"> - <i>Do nothing</i>: PWM output is not affected by PWM duty cycle value. - <i>Toggle at valley</i>: PWM output toggles at the beginning of PWM period. - <i>Toggle at peak</i>: PWM output toggles at the peak of PWM carrier wave. - <i>Toggle at peak/valley</i>: PWM output toggles at both of the peak/valley of PWM carrier wave. - <i>Set/Reset at valley/peak</i>: PWM output set to high at the valley, and reset to low at the peak of the carrier wave - <i>Reset/Set at valley/peak</i>: PWM output reset to low at the valley, and set to high at the peak of the carrier wave. - <i>Set at valley</i>: PWM output set to high at the beginning of PWM period. - <i>Reset at valley</i>: PWM output reset to low at the beginning of PWM period. - <i>Set at peak</i>: PWM output set to high at the peak of the carrier wave. - <i>Reset at peak</i>: PWM output reset to low at the peak of the carrier wave.

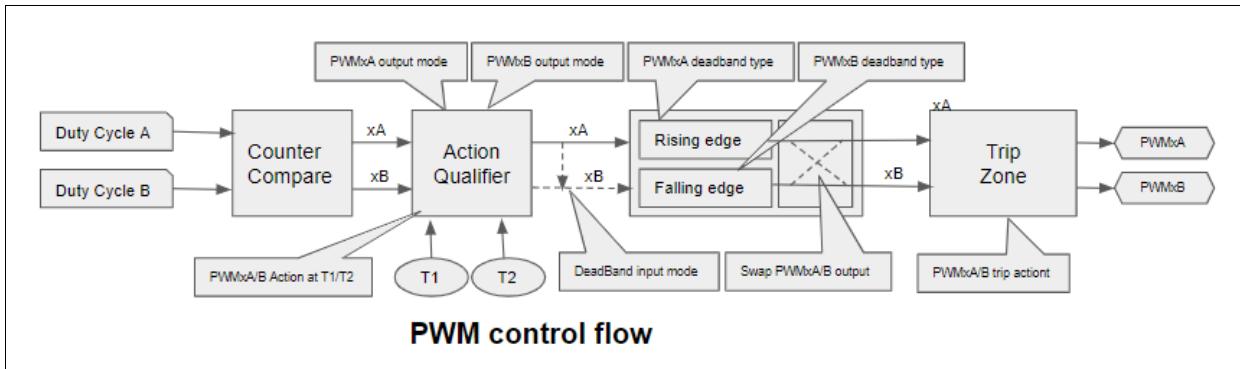
PWMxA Output Mode PWMxB Output Mode	<p>The followings are for PWMxA only:</p> <ul style="list-style-type: none"> - <i>Set high at CMPA</i>: PWM output starts low at the beginning of PWM period and turns high when CMPA is set. - <i>Set low at CMPA</i>: PWM output starts high at the beginning of PWM period and turns low when CMPA is set. - <i>Set high/low at CMPA/B</i>: PWM output starts low at the beginning of PWM period and turns high when CMPA is set; turns low when CMPB is set. - <i>Set low/high at CMPA/B</i>: PWM output starts high at the beginning of PWM period and turns low when CMPA is set; turns high when CMPB is set.
Trigger ADC	<p>The followings are for PWMxB only:</p> <ul style="list-style-type: none"> - <i>Complementary to PWMxA</i>: PWM output is complementary to PWMxA. - <i>Set high at CMPB</i>: PWM output starts low at the beginning of PWM period and turns high when CMPA is set. For PWMxA only. - <i>Set low at CMPB</i>: PWM output starts high at the beginning of PWM period and turns low when CMPA is set. For PWMxA only.
ADC Trigger Position	Setting whether for the PWM generator to trigger the A/D converter. It can be one of the following: <ul style="list-style-type: none"> - <i>Do not trigger ADC</i>: PWM does not trigger the A/D converter. - <i>Trigger ADC</i>: PWM will trigger the A/D converter.
T1 Source T2 Source	A/D trigger position ranges from 0 to a value less than 1. When it is 0, the A/D converter is triggered at the beginning of the PWM cycle, and when it is 0.5, the A/D converter is triggered at the 180° position of the PWM cycle.
PWMxA Action at T1(Up) PWMxA Action at T1(Down) PWMxB Action at T1(Up) PWMxB Action at T1(Down) PWMxA Action at T2(Up) PWMxA Action at T2(Down) PWMxB Action at T2(Up) PWMxB Action at T2(Down)	<p>T1 and T2 event sources for PWM trip actions, can be one of the followings:</p> <ul style="list-style-type: none"> - <i>Do not use</i>, - <i>DCAEVT1</i>, - <i>DCAEVT2</i>, - <i>DCBEVT1</i>, - <i>DCBEVT2</i>, - <i>TZ1</i>, - <i>TZ2</i>, - <i>TZ3</i>. <p>PWMxA (PWMxB) output can be forced at the following actions when T1 (T2) event occurs at PWM waveform up (down) count period:</p> <ul style="list-style-type: none"> - <i>Do nothing</i>, - <i>Clear</i>, - <i>Set</i>, - <i>Toggle</i>.

Dead Band Input Mode	Specify the dead band's input mode: <ul style="list-style-type: none"> - <i>Use both inputs</i>: PWM deadband sub-module's inputs are from both PWMxA and PWMxB - <i>Use PWMxA input only</i>: PWM dead band sub-module's inputs are from PWMxA only
PWMxA Dead Band Type	The type of PWM dead band, it can be one of the followings:
PWMxB Dead Band Type	<ul style="list-style-type: none"> - <i>No Delay</i>: No delay applied - <i>Rising/Falling edge delay</i>: Apply rising/falling edge delay to PWMxA/B - <i>Rising/Falling edge delay (invert)</i>: Apply rising/falling edge delay to PWMxA/B then invert it.
Swap PWMxA/B Output	Specify if PWM output A and B should be swapped
Rising edge Dead Time (us)	The dead time for the rising edge of the PWM generator, in us
Falling Edge Dead Time (us)	The dead time for the falling edge of the PWM generator, in us
Use Trip-Zone 1, 2, 3	Define whether the PWM generator uses the trip-zone signal or not. It can be one of the followings: <ul style="list-style-type: none"> - <i>Do not use</i>: Disable the i_{th} trip-zone signal. - <i>One shot</i>: The PWM generator uses the trip-zone signal in the one-shot mode. Once triggered, the PWM must be started manually. - <i>Cycle by cycle</i>: The PWM generator uses the trip-zone signal in the cycle-by-cycle basis. The trip-zone signal is effective within the current cycle, and PWM will automatically re-start in the next cycle.
Select Trip Events	Specify the combination of DC trip events applied to the current PWM. It can be one or any combination of DCAEVT1, DCAEVT2, DCBEVT1, and DCBEVT2.
Cycle-by-cycle Lock Clear	Specify when to clear the lock when in cycle-by-cycle mode: <ul style="list-style-type: none"> - <i>At PWM valley</i> - <i>At PWM valley/peak</i>.
PWMxA Trip Action(Up)	Define how the PWM generator responds to the trip action. It can be one of the following:
PWMxA Trip Action(Down)	<ul style="list-style-type: none"> - <i>High impedance</i>: PWM outputs in high impedance - <i>Force high</i>: Set PWM A high and B low. - <i>Force low</i>: Set PWM A low and B high. - <i>Toggle</i>: Toggle the output - <i>No action</i>: No action taken.
PWMxB Trip Action(Up)	
PWMxB Trip Action(Down)	
Peak Value	Peak value V_{pk} of the carrier wave. The lower end is 0, not -Vpk
Phase Shift	Phase shift of the output with respect to the reference PWM generator output, in deg. For example, when the phase shift is -30, the output will be shifted to the right (lagging) by 30 deg. with respect the reference PWM output.
Initial Input Value A	Initial value of the inputs A and B.
Initial Input Value B	

Use HRPWM	Define the high-resolution PWM. It can be one of the followings: <ul style="list-style-type: none"> - Do not use HRPWM - Use HRPWM without calibration - Use HRPWM with calibration
Start PWM at Beginning	When it is set to "Start", PWM will start right from the beginning. If it is set to "Do not start", one needs to start PWM using the "Start PWM" function block.

The 2-phase PWM generator generates two PWM signals with specific configurations for typical power converter applications. The PWM carrier wave is either sawtooth or triangular, it ranges from 0 to the peak value Vpk.

PWM module's **control flow** is as follows:



T1/T2 events are defined at F28004x Trip Zone block.

The PWM carrier wave is either sawtooth or triangular, it ranges from 0 to the peak value Vpk.

The following **PWMxA/B output mode combination** is allowed in SimCoder:

PWMxA output mode	PWMxB output mode	Sawtooth	Triangular
Toggle	Toggle		
	Set high at duty cycle 2	PwmxB set high at duty cycle 2	PwmxB set high/low at count up/down period at duty cycle 2.
	Set low at duty cycle 2	PwmxB sets high at duty cycle 2	PwmxB set low/high at count up/down period at duty cycle 2.
Set high at duty cycle 1	Toggle	PwmxA sets high at duty cycle 1	PwmxA sets high/low at count up/down period at duty cycle 1
	Set high at duty cycle 2	PwmxA/B set high at duty cycle 1/2	PwmxA set high/low at count up/down period at duty cycle 1; PwmxB set high/low at count up/down period at duty cycle 2.
	Set low at duty cycle 2	PwmxA/B set high/low at duty cycle 1/2	PwmxA set high/low at count up/down period at duty cycle 1; PwmxB set low/high at count up/down period at duty cycle 2.

Set low at duty cycle 1	Set high at duty cycle 2	PwmxA/B sets low/high at duty cycle 1/2	PwmxA sets low/high at count up/down period at duty cycle 1; PwmxB set high/low at count up/down period at duty cycle 2.
	Set low at duty cycle 2	PwmxA/B sets low at duty cycle 1/2	PwmxA sets low/high at count up/down period at duty cycle 1; PwmxB set low/high at count up/down period at duty cycle 2.
Set high/low at duty cycle 1/2	Toggle	PwmxA sets high/low at duty cycle 1/2	PwmxA sets high/low at count up/down period at duty cycle 1/2.
Set low/high at duty cycle 1/2	Toggle	PwmxA sets low/high at duty cycle 1/2	PwmxA sets high/low at count up/down period at duty cycle 1/2.

"Toggle" in the above table can be one of following modes:

- Do nothing, Toggle at valley,
- Toggle at peak, Toggle at valley/peak,
- Set/reset at valley/peak,
- Reset/set at valley/peak,
- Set at valley, Reset at valley,
- Set at peak and Reset at peak.

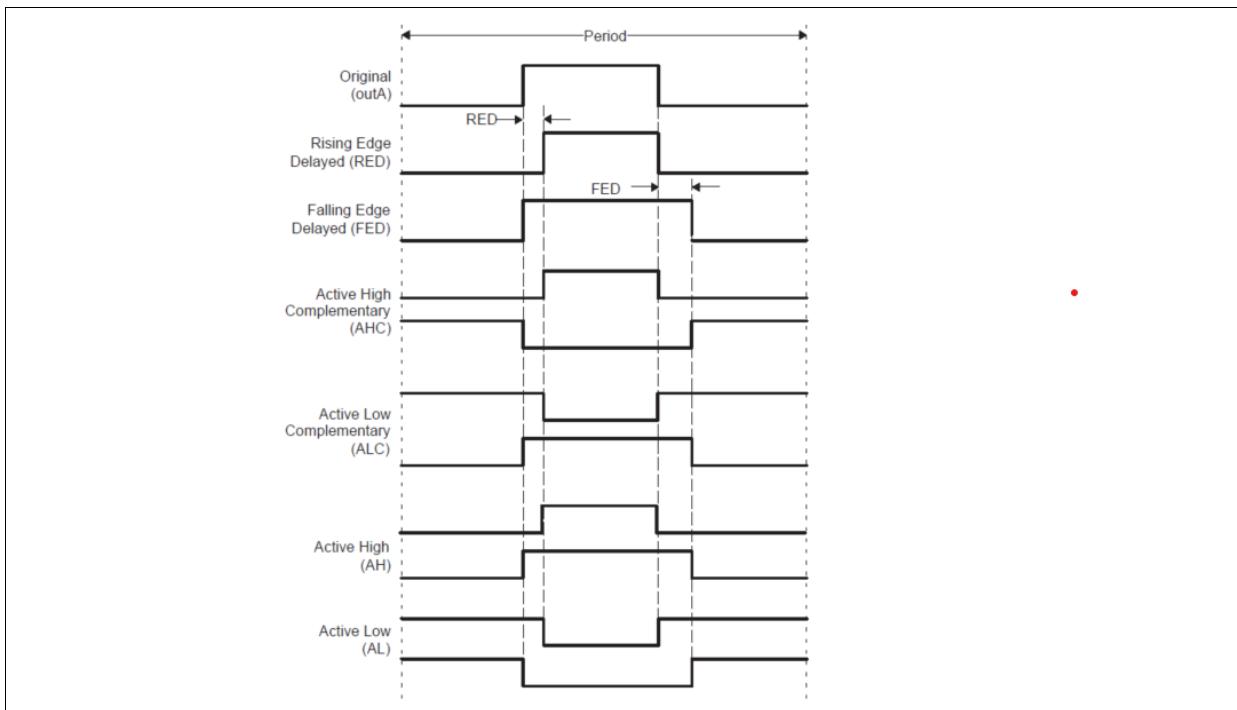
Dead band can be allowed only when PWMxB output mode is *Complementary to PWMxA*. Usually rising edge delay is set to PWMxA, falling edge delay is set to PWMxB, if rising edge delay is set to PWMxB and/or falling edge is set to PWMxA, Set parameter 'Swap PWMxA/B Output' to 'Yes'.

Dead band only applies to PWM actions specified at the following parameters:

PWMxA/B Output Mode, PWMxA/B Action at T1/2(Up/Down).

Dead band is not applied to PWM trip actions.

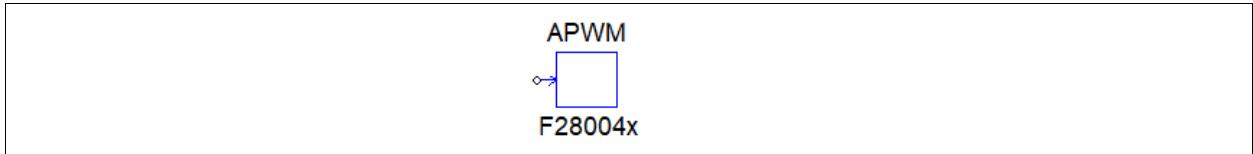
Dead band setting acts as the graph below.



11.4.3 Single PWM (shared with capture)

A single PWM (shared with capture) generator, also called APWM, shares the same resource as captures. It has restricted functionality as compared to other PWM generators. They cannot trigger the A/D converter and cannot use trip-zone signals. Also, because of the common resource, when a shared GPIO port is used for the capture, it can not be used for the PWM generator.

Image:



Attributes:

Parameters	Description
PWM Source	Source of the PWM generator. It can be any from APWM1 to APWM7
PWM Frequency	Frequency of the PWM generator, in Hz
Carrier Wave Type	The carrier wave type and the initial PWM output state. It can be one of the following: <ul style="list-style-type: none">- <i>Sawtooth (start low)</i>: Sawtooth wave, with the PWM output in the low state initially.- <i>Sawtooth (start high)</i>: Sawtooth wave, with the PWM output in the high state initially.
Stop Action	The output status when the PWM generator is stopped. It can be one of the following: <ul style="list-style-type: none">- <i>Output low</i>: The PWM output will be set to low.- <i>Output high</i>: The PWM output will be set to high.
Peak-to-Peak Value	Peak-to-peak value of the carrier wave
Offset Value	DC offset value of the carrier wave
Phase Shift	Phase shift of the output with respect to the reference PWM generator, in deg.
Initial Input Value	Initial value of the input
Start PWM at Beginning	When it is set to <i>Start</i> , PWM will start right from the beginning. If it is set to <i>Do not start</i> , one needs to start PWM using the Start PWM block.

Similar to 1-phase PWM generators, an APWM generator can generate a PWM signal that has a phase shift with respect to another PWM generator. The way how PWM blocks are defined for phase shift is explained in Section 11.4.4.

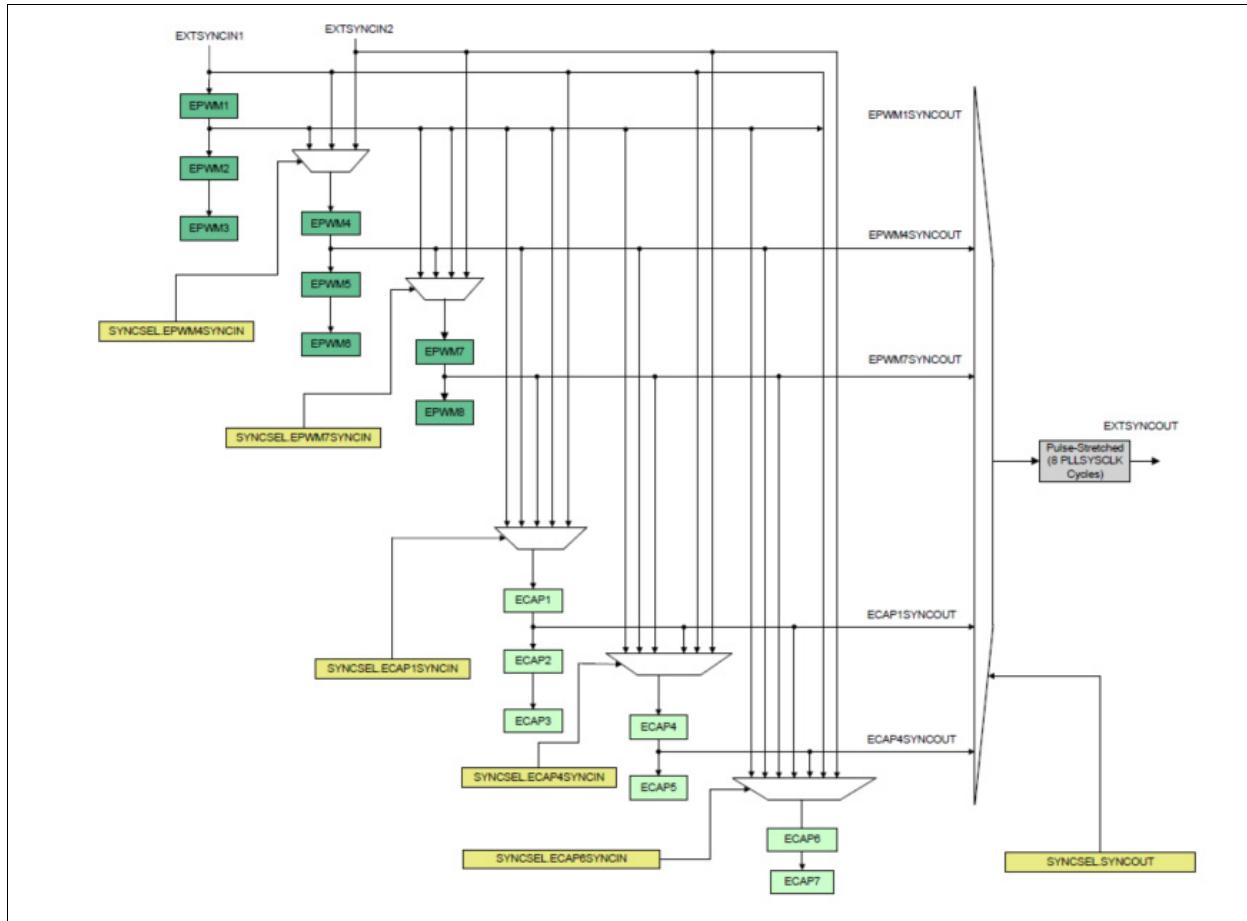
11.4.4 Synchronization Between PWM Blocks

Three types of PWM blocks can be synchronized, and phase shifts can be defined between each other:

- 1-phase PWM,
- 1-phase PWM (phase shift), and
- Single PWM (shared with capture) (APWM).

Any 1-phase PWM block can generate PWM signal that is phase shifted with respect to another PWM signal.

F28004x synchronization scheme is shown below:



- The master PWM can be EPWM1, EPWM4, EPWM7, APWM1, APWM4 and APWM6.
- Any subsequent EPWM or APWM can be slave block of the preceding EPWMs or APWMs in EPWM1, EPWM4, EPWM7, APWM1, APWM4 and APWM6 series.
- Any EPWM or APWM that is not master EPWM or APWM can be the slave if its close preceding EPWM or APWM block is a master EPWM or APWM.
- Any EPWM or APWM that is not master EPWM or APWM can be the slave if its preceding EPWM or APWM is a slave EPWM or APWM block

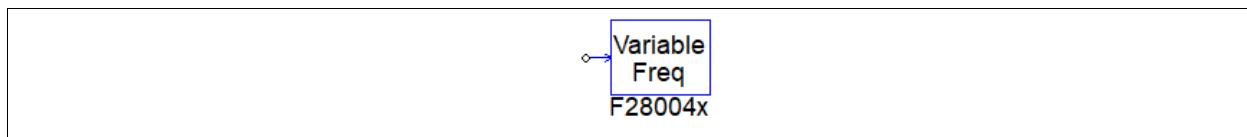
The correct phase shift sequences are as follows:

1. EPWM1, EPWM2, EPWM3
2. EPWM1, EPWM2, EPWM4, EPWM5, EPWM6
3. EPWM4, EPWM7, EPWM8
4. EPWM1, APWM1, APWM2, APWM3
5. EPWM7, APWM4, APWM5, APWM6, APWM7

11.5 Variable Frequency PWM

The Variable Frequency PWM block provides the function to change the sampling frequency of a PWM generator. The image and parameters are shown below.

Image:



Attributes:

Parameters	Description
PWM Source	Source of the PWM generator. It can be one listed in the pull-down menu.
Adjust Interrupt Pos.	Specify if the interrupt position is adjusted with the frequency. It can be one of the following: - <i>Do not adjust</i> : The interrupt position will remain unchanged as calculated with the base frequency. - <i>Adjust</i> : The interrupt position will be recalculated at the beginning of each cycle based on the new frequency.
Adjust Ramp Compensation	Specify if the ramp compensation of the comparator DAC block is adjusted with the frequency. It can be one of the following: - <i>Do not adjust</i> : The ramp compensation will remain unchanged as calculated with the base frequency. - <i>Adjust</i> : The ramp compensation will be recalculated at the beginning of each cycle based on the new frequency.

The sampling frequency of the corresponding PWM block will be changed at the beginning of the next PWM period as follows:

$$\text{PWM_Frequency} = \text{PWM_Base_Frequency} / \text{Input_Value}$$

where *PWM_Base_Frequency* is the sampling frequency of the corresponding PWM block, and *Input_Value* is the input value of this block.

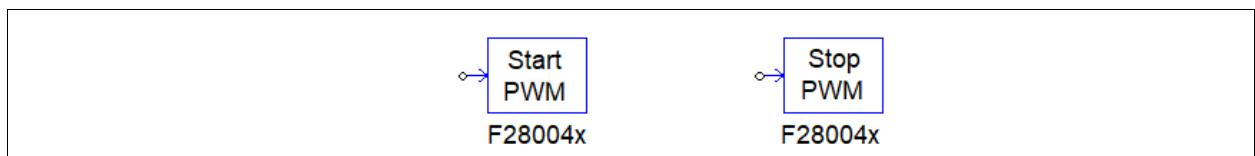
If the interrupt position is to be adjusted, the interrupt position will be recalculated in each cycle. Since adjusting the interrupt position takes time, if the frequency change is small, it is recommended not to adjust the interrupt position.

Similarly, if the ramp compensation is to be adjusted, the ramp compensation will be recalculated in each cycle. Since adjusting the ramp compensation takes time, if the frequency change is small, it is recommended not to adjust the ramp compensation.

11.6 Start PWM and Stop PWM

The Start PWM and Stop PWM blocks provide the function to start/stop a PWM generator. The images and parameters are shown below.

Image:



Attributes:

Parameters	Description
PWM Source	The source of the PWM generator. It can be any of PWM blocks or captures in the pull-down list

11.7 PWM Trip-Zone And Trip-Zone State

All the F28004x's PWM DC (Digital Compare) trip-zone signals come from the PWM X-Bar. The PWM Trip-zone block is used to handle external fault or trip conditions. The corresponding PWM outputs can be programmed to respond accordingly.

The Trip-Zone State block indicates the type of the trip-zone signal, whether it is one-shot or cycle-by-cycle.

11.7.1 PWM Trip-Zone

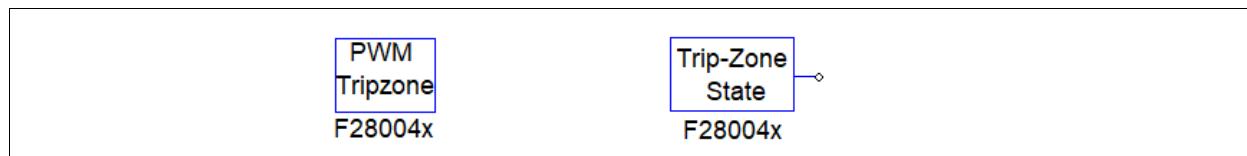
Any PWM X-Bar signal can be used in one or multiple PWMs. The interrupt generated by trip-zone signals are handled by the interrupt block.

Any trip-zone signals though Digital Compare trigger a trip action in the specified active level (high or low).

The trip-zone interrupt can be generated in either one-shot mode or cycle-by-cycle mode, as defined in the PWM generator parameter input. In the cycle-by-cycle mode, the interrupt only affects the PWM output within the current PWM cycle. On the other hand, in the one-shot mode, interrupt will set the PWM output permanently, and the PWM generator must be restarted to resume the operation.

Valley switch is supported. When simulating valley switching circuit, one should set a proper simulation step according to the specified trip-zone signals, usually a smaller time step is recommended. If trip-zone signal changes multiple times in one time step, SimCoder could not simulate the circuit correctly.

Image:



Attributes for Trip-Zone:

Parameters	Description
PWM Source DC Trip Source1(DCAH) DC Trip Source2(DCAL) DC Trip Source1(DCBH) DC Trip Source2(DCBL)	Source of the PWM generator. It can be any of the PWM blocks in the pull-down list. Specify the digital compare (DC) trip source DCAH for PWMxA. The PWM channel may have up to two DC trip sources: DCAH and DCAL. The trip source can be one of the following: <ul style="list-style-type: none">- Do not use: PWM doesn't use this signal.- TZ1(T1): PWM uses trip-zone 1 as digital compare input signal.- TZ2(T2): PWM uses trip-zone 2 as digital compare input signal.- TZ3(T3): PWM uses trip-zone 3 as digital compare input signal.- TRIP4(T4): PWM uses TRIP4 as digital compare input signal.- TRIP5(T5): PWM uses TRIP5 as digital compare input signal.- TRIP7(T7): PWM uses TRIP7 as digital compare input signal.- TRIP7(T8): PWM uses TRIP8 as digital compare input signal.- TRIP7(T9): PWM uses TRIP9 as digital compare input signal.- TRIP10(T10): PWM uses TRIP10 as digital compare input signal.- TRIP11(T11): PWM uses TRIP11 as digital compare input signal.- TRIP12(T12): PWM uses TRIP12 as digital compare input signal.- T1 - T5- T7 - T12- T5 T8 T11.

One Shot (DCAEVT1)	Define how the one-shot signal is used for the DC trip signal of PWMA.
Cycle By Cycle (DCAEVT2)	The active level of the DC trip signal can be selected from the following:
One Shot (DCBEVT1)	<ul style="list-style-type: none"> - <i>Do not use</i>: PWM doesn't use this signal. - <i>Trip source 1 is low</i>: PWM is tripped if the source signal 1 is low. - <i>Trip source 1 is high</i>: PWM is tripped if the source signal 1 is high. - <i>Trip source 2 is low</i>: PWM is tripped if the source signal 2 is low. - <i>Trip source 2 is high</i>: PWM is tripped if the source signal 2 is high. - <i>Trip source 1 low & source 2 high</i>: PWM is tripped if the source 1 signal is low and at the same time source 2 signal is high.
Cycle By Cycle (DCBEVT2)	
DC Event Filter Source	Source of the digital compare event filter. It can be one of the following: <ul style="list-style-type: none"> - <i>Do not use</i>: Do not use DC event filter. - <i>DCAEVT1</i>: Use DCAEVT1 as the filter source. - <i>DCAEVT2</i>: Use DCAEVT2 as the filter source. - <i>DCBEVT1</i>: Use DCBEVT1 as the filter source. - <i>DCBEVT2</i>: Use DCBEVT2 as the filter source.
Blanking Window Start From	The Blanking Window may start in the following positions: <ul style="list-style-type: none"> - <i>Beginning of PWM period</i> - <i>Beginning/Middle of PWM period</i>
Blanking Window Width (us)	Width of the blanking window, in us. The width is limited by the hardware, and can be calculated as below: $16383 / \text{PWM Clock Frequency}$ <p>For example, when CPO speed is 100MHz, PWM clock is 100MHz, the width range is 0 to 163.83us.</p>
Blanking Window Range	Specify how the blanking window is applied. It can be one of the following: <ul style="list-style-type: none"> - <i>In the window</i>: The blanking action is applied with the window defined (from the start position for a window width defined) - <i>Out of window</i>: The blanking action is applied outside the window defined.
Applying Event Filtering	Specify how event filtering is applied to digital compare events. The event filtering can be applied to any one or any combinations of the digital compare events: <ul style="list-style-type: none"> - DCAEVT1, - DCAEVT2, - DCBEVT1, - DCBEVT2
Enable Valley Switching	Specify the DC event that is used for valley switch signal, one of the following DC events can be chose: <ul style="list-style-type: none"> - Do not use, - DCAEVT1, - DCAEVT2, - DCBEVT1, - DCBEVT2

Trigger Event Selection	Specify the chosen trigger event, PWM starts to count up when this trigger event meets the edge specified by " Trigger Event Edge Type ". one of the following DC events can be chose: <ul style="list-style-type: none"> - Beginning of carrier wave - Halfway of carrier wave - DCAEVT1, - DCAEVT2, - DCBEVT1, - DCBEVT2
Trigger Event Edge Type	Specify trigger event edge type. It can be one of the followings: <ul style="list-style-type: none"> - Rising edge - Falling edge - Rising/Falling edge.
Trigger Event Edge Count	Specify trigger event edge count, PWM starts to count filter event edge when this trigger event counter equals to this value.
Filter Event Start Edge	Specify the start edge count for the specified filter event. The filter event is chosen by " DC Event Filter Source ".
Filter Event Stop Edge	Specify the stop edge count for the specified filter event. The filter event is chosen by " DC Event Filter Source ".
Software Delay Time (us)	Specify software delay time, in us.
Delay Calculate Mode	<p>Specify delay time calculate mode. When the condition specified by parameter 'Trigger Event Edge Count' meets, the specified filter event passes through after the delay time decided by this parameter.</p> <p>Valley delay below is the interval between Filter Event Start Edge and Filter Event Stop Edge.</p> <p>The calculation mode of delay time can be one of the follows:</p> <ul style="list-style-type: none"> - No delay time: Delay time is 0. - Use software delay: Delay time is software delay time. - Use Software delay + valley delay: Delay time is software delay time + valley delay. - Use Software delay + valley delay/2: Delay time is software delay time + valley delay / 2. - Use Software delay + valley delay/4: Delay time is software delay time + valley delay / 4. - Use Software delay + valley delay/16: Delay time is software delay time + valley delay / 16.

Note: when defining the interrupt block associate with trip-zone, the "Device Name" parameter of the interrupt block should be the name of the PWM generator, not the trip-zone block name. For example, if a PWM generator called "PWM_G1" uses trip-zone 1 in the trip-zone block "TZ1". The "Device Name" of the corresponding interrupt block should be "PWM_G1", not "TZ1". The "Channel Number" parameter in the interrupt block is not used in this case.

The trip-zone interrupt can be generated in either one-shot mode or cycle-by-cycle mode, as defined in the PWM generator parameter input. In the cycle-by-cycle mode, the interrupt only affects the PWM output within the current PWM cycle. On the other hand, in the one-shot mode, interrupt triggers a trip action when the input signal is low (0). will set the PWM output permanently, and the PWM generator must be restarted to resume the operation.

11.7.2 Trip-Zone State

The Trip-Zone State element indicates whether the trip-zone signal is in one-shot mode or cycle-by-cycle mode when it triggers a PWM generator to generate an interrupt.

Attributes for Trip-Zone State:

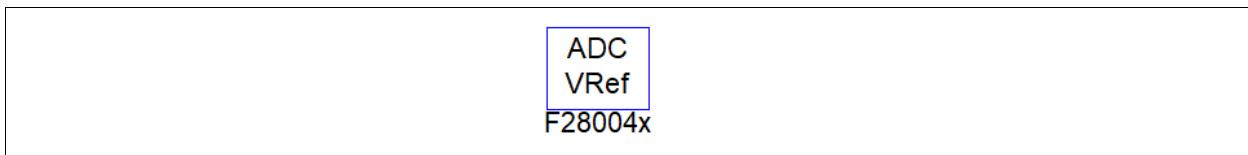
Parameters	Description
PWM Source	Source of the PWM generator. It can be any one in the pull-down list

The trip-zone state element is usually used in the trip-zone interrupt routine to indicate the operation mode of the trip-zone signal. When the output of this element is 1, it indicates that the trip-zone signal is in one-shot mode. When the output is 0, the trip-zone signal is in cycle-by-cycle mode.

11.8 ADC Voltage Reference

The Voltage Reference for A/D converters, D/A converters, and comparators.

Image:



Attributes:

Parameters	Description
ADC-A Vol. Ref. Usage	Specify the voltage reference to be used for each A/D converter,: External VRef
ADC-B/C Vol. Ref. Usage	Internal VRef (1.65V) Internal VRef (2.5V)
ADC-A High Voltage Ref.	Specify the external high voltage reference for ADC. The value must be in the range of 2.4V to 3.3V.
ADC-B/C High Voltage Ref.	
Use ADC-B3 as VDAC	Specify if ADC-B3 is used as VDAC.
DAC Voltage Ref	Specify the voltage reference value of VDAC.

All high voltage reference (include VDAC) can be set to 2.4V to 3.3V.

If "Use ADC-B3 as VDAC" is set as Yes, DACs and Comparator DACs can use VDAC as voltage reference. Otherwise, VDAC can not be selected by DACs and Comparator DACs. PSIM does not need ADC-B3 to connect a voltage reference in schematic, but in target system, ADC-B3 must connect a voltage reference.

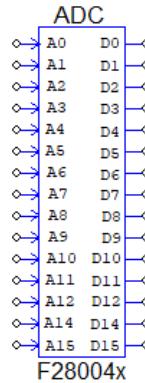
11.9 A/D Converter

F28004x target provides 16-channel 12-bit A/D converter. All channels in each ADC module share the same sample-and-hold (s/h) circuit. The interrupt priority can be either PIE Group1 or PIE Group10. Each A/D channel can accept multiple trigger sources, and each interrupt source can trigger multiple A/D channels.

An A/D converter can be triggered by one or more of the PWM generators, by Timer1 or by Timer2. In a schematic, if an A/D channel is not associated with a PWM, one should connect a ZOH block at the output of the A/D channel so that SimCoder will use the selected timer as trigger source.

The image and the parameters of the A/D converter in the SimCoder library for F28004x target are described below.

Image:



Attributes:

Parameters	Description
ADC Source	Specify ADC source, it can be one of ADC-A, ADC-B and ADC-C.
Chn i Gain	Gain of the i_{th} A/D converter channel, where i ranges from 0 to 15.
Chn i Sample Time (us)	Specify the sample time of the A/D converter channel i , where i ranges from 0 to 15.
Conversion Order	<p>Order of the A/D conversion. If the field is left blank (undefined), the conversion will be done based on the sequential numbers of the A/D channel. ADC channel name used in this parameter is as follows: A0, A1, A2, A3, A4, A5, A14, A15.</p> <p>For example, if A0, A2, A4, A14, and A15 are used, the conversion will be done in this order: A0, A2, A4, A14, and A15. If you wish certain channels to be performed first, you can define the order here. For example, if the conversion order is defined as:</p> <p style="text-align: center;">A4,A0,A2,A14,A15</p> <p>The conversion will be done in the order defined, that is, A4 first, then A0, then A2, and so on.</p>
High Priority PIE Selection	Specify if the highest priority interrupt uses <i>PIE Group1</i> or <i>PIE Group10</i> .

The output of the A/D converter is scaled based:

$$V_o = k * V_i$$

where V_i is the value at the input port of the A/D converter.

If there is an ADC converter block in the schematic, a voltage reference block is needed, too.

If ADC uses internal +1.65V voltage reference, the real voltage reference for ADC should be +3.3V.

For 100-pin CPU package, ADC-A uses VREFHIA/VREFLOA, ADC-B and ADC-C use VREFHIB/VREFLOB. For 64-pin or 56-pin CPU package, there is only VREFHIA/VREFLOA, ADC-A, ADC-B and ADC-C use the same external voltage reference.

In the three ADC converter blocks in F28004x, each block has up to 16 channels. Some ADC channels are not linked to DSP pin for external inputs. They are only internally link to PGA outputs or inner temperature sensors. Some ADC input channels uses same pin with PGA, CMPSS (comparator) and DAC.

The table below shows the details for the ADC subsystem:

PIN NAME	GROUP NAME	PACKAGE			ALWAYS CONNECTED (NO MUX)				COMPARATOR SUBSYSTEM (MUX)				AIO INPUT	
		100 PZ	64 PM	56 RSH	ADCA	ADCB	ADCC	PGA	DAC	HIGH POSITIVE	HIGH NEGATIVE	LOW POSITIVE	LOW NEGATIVE	
Analog Group 1														
A3	G1_ADCAB	10			A3					CMP1_HP3	CMP1_HN0	CMP1_LP3	CMP1_LN0	AIO233
A2/B6/PGA1_OF	PGA1_OF	9	9	8	A2	B6		PGA1_OF		CMP1_HP0		CMP1_LP0		AIO224
C0	G1_ADCC	19		12	10			C0		CMP1_HP1	CMP1_HN1	CMP1_LP1	CMP1_LN1	AIO237
PGA1_IN	PGA1_IN	18						PGA1_IN						
PGA1_GND	PGA1_GND	14	10	9				PGA1_GND						
-	PGA1_OUT ⁽¹⁾				A11	B7		PGA1_OUT		CMP1_HP4		CMP1_LP4		
Analog Group 2														
A5	G2_ADCAB	35			A5					CMP2_HP3	CMP2_HN0	CMP2_LP3	CMP2_LN0	AIO234
A4/B8/PGA2_OF	PGA2_OF	36	23	21	A4	B8		PGA2_OF		CMP2_HP0		CMP2_LP0		AIO225
C1	G2_ADCC	29		18	16			C1		CMP2_HP1	CMP2_HN1	CMP2_LP1	CMP2_LN1	AIO238
PGA2_IN	PGA2_IN	30						PGA2_IN						
PGA2_GND	PGA2_GND	32	20	18				PGA2_GND						
-	PGA2_OUT ⁽¹⁾				A12	B9		PGA2_OUT		CMP2_HP4		CMP2_LP4		
Analog Group 3														
B3/V/DAC	G3_ADCAB	8	8	7		B3			V/DAC	CMP3_HP3	CMP3_HN0	CMP3_LP3	CMP3_LN0	AIO242
B2/C6/PGA3_OF	PGA3_OF	7	7	6		B2	C6	PGA3_OF		CMP3_HP0		CMP3_LP0		AIO226
C2	G3_ADCC	21					C2			CMP3_HP1	CMP3_HN1	CMP3_LP1	CMP3_LN1	AIO244
PGA3_IN	PGA3_IN	20		13	11			PGA3_IN						
PGA3_GND	PGA3_GND	15	10	9				PGA3_GND						
-	PGA3_OUT ⁽¹⁾					B10	C7	PGA3_OUT		CMP3_HP4		CMP3_LP4		
Analog Group 4														
B5	G4_ADCAB				B5					CMP4_HP3	CMP4_HN0	CMP4_LP3	CMP4_LN0	AIO243
B4/C8/PGA4_OF	PGA4_OF	39	24	22		B4	C8	PGA4_OF		CMP4_HP0		CMP4_LP0		AIO227
C3	G4_ADCC						C3			CMP4_HP1	CMP4_HN1	CMP4_LP1	CMP4_LN1	AIO245
PGA4_IN	PGA4_IN							PGA4_IN						
PGA4_GND	PGA4_GND	32	20	18				PGA4_GND						
-	PGA4_OUT ⁽¹⁾					B11	C9	PGA4_OUT		CMP4_HP4		CMP4_LP4		
Analog Group 5														
A7	G5_ADCAB				A7					CMP5_HP3	CMP5_HN0	CMP5_LP3	CMP5_LN0	AIO235
A6/PGA5_OF	PGA5_OF	6	6		A6			PGA5_OF		CMP5_HP0		CMP5_LP0		AIO228
C4	G5_ADCC	17					C4			CMP5_HP1	CMP5_HN1	CMP5_LP1	CMP5_LN1	AIO239
PGA5_IN	PGA5_IN	16		11				PGA5_IN						
PGA5_GND	PGA5_GND	13	10	9				PGA5_GND						
-	PGA5_OUT ⁽²⁾					A14		PGA5_OUT		CMP5_HP4		CMP5_LP4		
Analog Group 6														
A9	G6_ADCAB	38			A9					CMP6_HP3	CMP6_HN0	CMP6_LP3	CMP6_LN0	AIO236
A8/PGA6_OF	PGA6_OF	37			A8			PGA6_OF		CMP6_HP0		CMP6_LP0		AIO229
C5	G6_ADCC						C5			CMP6_HP1	CMP6_HN1	CMP6_LP1	CMP6_LN1	AIO240
PGA6_IN	PGA6_IN							PGA6_IN						
PGA6_GND	PGA6_GND	32	20	18				PGA6_GND						
-	PGA6_OUT ⁽²⁾					A15		PGA6_OUT		CMP6_HP4		CMP6_LP4		
Analog Group 7														
B0	G7_ADCAB	41			B0					CMP7_HP3	CMP7_HN0	CMP7_LP3	CMP7_LN0	AIO241
A10/B1/C10/PGA7_OF	PGA7_OF ⁽³⁾	40	25	23	A10	B1	C10	PGA7_OF		CMP7_HP0		CMP7_LP0		AIO230
C14	G7_ADCC	44					C14			CMP7_HP1	CMP7_HN1	CMP7_LP1	CMP7_LN1	AIO246
PGA7_IN	PGA7_IN	43						PGA7_IN						
PGA7_GND	PGA7_GND	42						PGA7_GND						
-	PGA7_OUT ⁽²⁾					B12	C11	PGA7_OUT		CMP7_HP4		CMP7_LP4		
Other Analog														
A0/B15/C15/DACA_OUT		23	15	13	A0	B15	C15		DACA_OUT					AIO231
A1/DACB_OUT		22	14	12	A1		C12		DACB_OUT					AIO232
C12														AIO247
-	TempSensor ⁽²⁾					B14								

In this table:

(1)(2) Internal connection only, does not come to a device pin.

(3) PGA functionality not available on 64-pin or 56-pin packages.

SimCoder allows one input signal used by multiple devices, for example, from the table above, there is a pin named A2/B6/PGA1_OF, that means PGA1_OF has internally connected to ADCA2 and ADCB6. If the schematic uses PGA1 then ADCA2 and/or ADCB6 can be use PGA1_OF by connecting PGA1_OF to ADCA2 and/or ADCB6. Since PGA1_OF connects to PGA1 output through a resistor, SimCoder doesn't report an error message if ADCA2 and/or ADCB6 use another signal other than PGA1_OF, but ADCA2 and ADCB6 should use the same input signal if the schematic uses both channels.

Some ADC input channels are connected to PGAx_OUT internally, one can only connect PGAx_OUT to the ADC channel, connect any other signal to this kind of ADC channels results an error message reported on SimCoder.

Each ADC converter has up to 4 interrupts, one is in PIE1 with the highest interrupt priority, other 3 are in PIE10 with lower interrupt priority.

If any sampling rate uses any ADC channels, SimCoder assigns an ADC interrupt for the sampling rate. If setting parameter 'High Priority PIE Selection' to PIE group 1, SimCoder assigns the highest sampling rate to PIE1 ADC interrupt, the other lower sampling rates use PIE group 10; If the parameter 'High Priority PIE Selection' is set to PIE group 10, SimCoder assigns the all sampling rates related to this ADC converter in PIE10.

group 10.

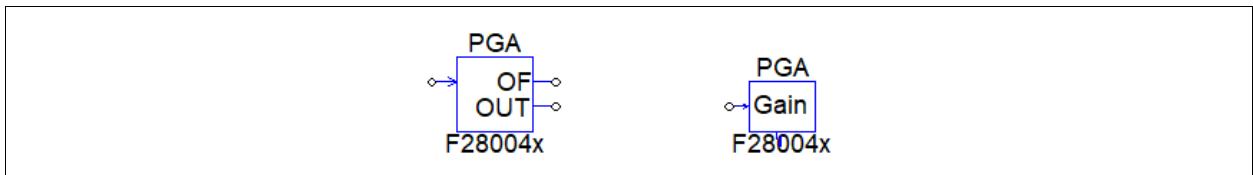
If the same sampling rate uses in multiple ADC converters, SimCoder checks the total conversion time on this sampling rate for all ADC converters, and sets the ADC converter with the longest conversion time to cause interrupt, the other ADC converters do not need to cause interrupt. In this way all related conversions are done when entering interrupt.

11.10 PGA and PGA Gain

In F28004x, there are seven PGAs (Program able Gain Amplifier). They share the same ports with ADC input channels and Comparators. Please check the connections between PGA, ADCs and comparators as shown in the table in section **10.9 A/D Converter**. PSIM will report an error if there is any incorrect connection between these devices.

The gain of these PGAs can be controlled in real time.

Images:



Attributes of PGA:

Parameters	Description
PGA Source	Specify PGA source, it can be one of the followings: PGA1, PGA2, ..., PGA7.
Default Gain	Specify PGA's default gain. The options are: 3 times, 6 times, 12 times and 24 times.
Filter Usage	Specify the resistor between PGA output and PGAx_OF. The options are: Disabled, 200ohm, 160ohm, 130ohm, 100ohm, 80ohm and 50ohm.

Attributes for PGA Gain:

Parameters	Description
PGA Source	Specify PGA source, it can be one of the followings: PGA1, PGA2, ..., PGA7.

For each PGA, the voltage reference is VDDA(+3.3V). Each PGA has 2 outputs, one named PGAx_OUT and it is internally connected to ADC input channel, or comparator input; another is PGAx_OF which shares the same pin with the specific channels of ADC and Comparator.

The gain of a PGA can be controlled but the PGA Gain block. The input and the gain relationship is defined as:

Input	Gain
0 or less	3 times
1	6 times
2	12 times
3 or greater	24 times

11.11 Comparator

In F28004x, there are seven comparators. They share the same ports with ADC input channels. PSIM will report an error if a port is defined as a comparator input but is used as a A/D converter input.

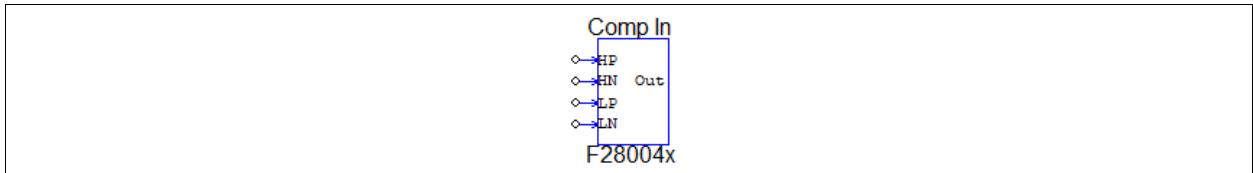
11.11.1 Comparator Input

In F28004x, there are 7 comparators. They share the same ports with ADC input channels and PGA. Each comparator has two of side comparators: high side comparator and low side comparator. For each side comparator, the positive signal is from an external analog input or other devices in the CPU, the negative signal is from an external analog input or the output of comparator inner DAC.

Please check the connections between comparators, ADCs and PGAs as shown in the table in section **10.9 A/D Converter**. PSIM will report an error if there is any incorrect connection between these devices.

Only one function can be designated for each port. Simcoder will report error if a port is defined as comparator input but is also used as a A/D converter channel in the same PSIM circuit schematic.

Image:



Attributes

Parameters	Description
Comparator Source	Specify Comparator source, it can be one of the followings: Comparator1, Comparator2, ..., Comparator7.
High Side Positive Input	Specify Comparator high side positive source. The options are: Do not use, CMPx_HP0(PGAx_OF) CMPx_HP1(Gx_ADCC) CMPx_HP2(PGAx_IN) CMPx_HP3(Gx_ADCAB) CMPx_HP4(PGAx_OUT)
High Side Negative Input	Specify Comparator high side negative source. The options are: Use inner DAC CMPx_HN0(Gx_ADCAB) CMPx_HN1(Gx_ADCC)
Low Side Positive Input	Specify Comparator high side positive source. The options are: Do not use, CMPx_LP0(PGAx_OF) CMPx_LP1(Gx_ADCC) CMPx_LP2(PGAx_IN) CMPx_LP3(Gx_ADCAB) CMPx_LP4(PGAx_OUT)
Low Side Negative Input	Specify Comparator high side negative source. The options are: Use inner DAC CMPx_LN0(Gx_ADCAB) CMPx_LN1(Gx_ADCC)
PWM Sync. Source	Specify the synchronization PWM of this comparator. The PWM options are in the pull-down list.

PWM's Sync. Pos (us)	Specify the position of PWM synchronization for this comparator, in us. The comparator uses the latest input value at the Comparator DAC when it receives the synchronization signal.
PWM Blank Source	Specify a PWM of which the PWM blank window definition will be applied. The options are in the pull-down list.
Input Hysteresis Selection	Specify the type of hysteresis applied for input signal. This information is ignored in simulation. It can be one of the followings: - <i>None</i> : No hysteresis applied. - <i>Typical hysteresis</i> : Typical hysteresis applied. - <i>2x hysteresis</i> : 2 times of hysteresis applied. - <i>3x hysteresis</i> : 3 times of hysteresis applied. - <i>4x hysteresis</i> : 4 times of hysteresis applied
High Output Invert	Specify if inverting the output of high/low side comparator.
Low Output Invert	
High Output Sync.	Specify if synchronization mode of the output of high/low side comparator. It can be one of the followings: - <i>Asynchronous</i> : No synchronization applied. - <i>Sync. by SysClk</i> : Input signal is synchronized by system clock. - <i>Digital filter result</i> : Input signal is filtered. - <i>Latched result</i> : Input signal is filtered and is latched when it becomes active.
Low Output Sync,	
High Filter Width (us)	Specify the filter window width for high/low side comparator, in us. This information is valid only when 'High Output Sync.' is selected as 'Digital filter result' or 'Latched result'.
Low Filter Width (us)	
High Threshold (%)	Specify the filter threshold for high/low side comparator, in %. This information is valid only when 'High Output Sync.' is selected as 'Digital filter result' or 'Latched result'.
Low Threshold (%)	
High Threshold (%)	Specify the filter threshold for High side or Low side comparator, in %. This information is valid only when 'High/Low Output Sync.' is set as 'Digital filter result' or 'Latched result'.
Low Threshold (%)	

For example, assume that one needs to specify comparator 1's high side positive signal, From F28004x CompIn block, there are five choices for this input, they are

- CMPx_HP0(PGAx_OF),
- CMPx_HP1(Gx_ADDCC),
- CMPx_HP2(PGAx_IN),
- CMPx_HP3(Gx_ADCAB) and
- CMPx_HP4(PGAx_OUT).

Since this is comparator 1, one can choose high side positive signal from CMP1_HP0, CMP1_HP1, CMP1_HP2, CMP1_HP3, CMP1_HP4. From the table about, we know:

- CMP1_HP0 is PGA1_OF,
- CMP1_HP1 is ADCC0,
- CMP1_HP2 is PGA1_IN,
- CMP1_HP3 is ADCA3
- and CMP1_HP4 is PGA1_OUT.

Here if one choose CMP1_HP0(PGA1_OF), comparator 1's high positive should connect to PGA1's PGA1_OF signal, otherwise SimCoder will report error message; if one choose CMP1_HP4(PGA1_OUT), comparator 1's high positive should connect to PGA1's PGA1_OUT signal, otherwise SimCoder will report error message.

Note: For F28004x 100-pin package, CMPx_HP1 and CMPx_HP2 are different pins except comparator 6, but for other packages, CMPx_HP1 and CMPx_HP2 are connected together in the CPU.

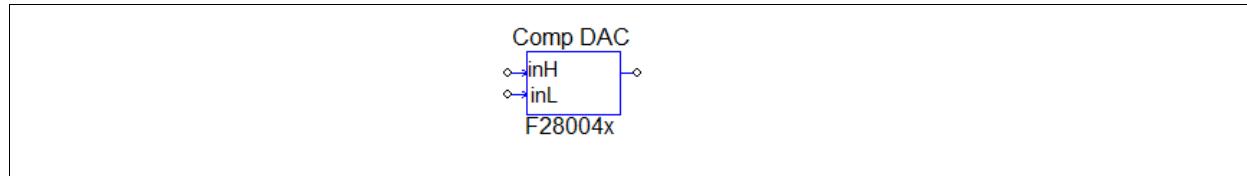
SimCoder allows one input signal used by multiple devices, for example, from the table above, there is a pin

named A2/B6/PGA1_OF, that means PGA1_OF has internally connected to ADCA2 and ADCB6. If the schematic uses PGA1 then ADCA2 and/or ADCB6 can be used by connecting PGA1_OF to ADCA2 and/or ADCB6. Since PGA1_OF connects to PGA1 output through a resistor, SimCoder doesn't report an error message if ADCA2 and/or ADCB6 use another signal other than PGA1_OF, but ADCA2 and ADCB6 should use the same input signal if the schematic uses both channels.

11.11.2 Comparator DAC

Each comparator DAC has two 12-bit D/A converter. Outputs of side comparator DAC are linked to the corresponding inverting inputs (Input B) of the corresponding side comparator. Also, a comparator DAC block cannot be used alone, it must be used with a comparator input block and they have the same comparator source.

Image:



Attributes:

Parameters	Description
Comparator Source	Comparator source, it can be any of the 7 comparators.
DAC Voltage Reference	Voltage reference of inner DACs, it can be either 'Use VDDA(3.3V)' or 'Use VDAC'.
DACH Usage	The other signal of the high side comparator: <ul style="list-style-type: none"> - <i>Do not use DACH</i>: The high side inner DAC is not used, the other signal of high side comparator is Input B if the high side comparator is used. - <i>Use constant DAC value</i>: High side inner DAC is used but the DAC output is a constant. The generated program only sets this constant value in the initialization. - <i>Use variable DAC value</i>: High side inner DAC is used and the generated program sets inner DAC value according to its sampling frequency. - <i>Use ramp(Constant)</i>: High side inner DAC is set by ramp generator, the maximum value of ramp generator is a constant. The generated program only sets this constant value in the initialization. - <i>Use ramp(Variable)</i>: High side inner DAC is set by ramp generator, the generated program sets the maximum value of ramp generator according to its sampling frequency.
DACH Range	The input range of high side inner DAC.
Initial DACH Value	Initial value of high side inner DAC. This value is set at the initialization to DACH/Ramp if DACH/Ramp only uses a constant value.
Total Ramp Compensation	Total compensation of the ramp generator in one PWM period. It represents the total decrease of the ramp in one cycle.
Ramp Delay (us)	The delay time in the beginning of PWM period, Ramp decreases when delay time is over, in us.

DACL Usage	Specify the another signal of low side comparator. It can be one of the following: <ul style="list-style-type: none"> - <i>Do not use DACL</i>: The low side inner DAC is not used, the other signal of high side comparator is Input B if the low side comparator is used. - <i>Use constant DAC value</i>: Low side inner DAC is used but the DAC output is a constant. The generated program only sets this constant value in the initialization. - <i>Use variable DAC value</i>: Low side inner DAC is used and the generated program sets inner DAC value according to its sampling frequency.
DACL Range	The input range of low side inner DAC.
Initial DACL Value	Initial value of low side inner DAC. This value is set at the initialization to DACL if DACL only uses a constant value.

Ramp generator only exists in high side inner DAC. When ramp generator is used, a synchronization PWM should be specified in the corresponding Comparator Input block. The input value is applied to the maximum value of ramp generator, ramp generator decreases after the delay time from the beginning of the sync. PWM period is over. When high side comparator's result becomes high, ramp generator restore ramp value to the maximum value and stop decrease, so the active result only remain a short time. It's better to choose 'Latched result' for parameter 'High Output Sync.' in Comparator Input block. The latched result resets at the beginning of the next PWM period.

If the inner DAC is used, the input value is applied to DAC output immediately when there is no sync. PWM specified in Comparator Input block; otherwise the input value is applied to DAC output at the beginning of the sync. PWM's next period.

The voltage reference of inner DAC can be VDDA(3.3V) or VDAC. If VDAC is chose, ADC-B0 is used as VDAC. In Psim schematic, this should be only specified in Voltage Reference block, no need to link a voltage source to ADC-B0. But in target hardware, a voltage reference should be linked to ADC-B0. The output can be calculated as follows:

Use VDDA:

$$DAC_Output = DAC_Input * 3.3 / DAC_Range$$

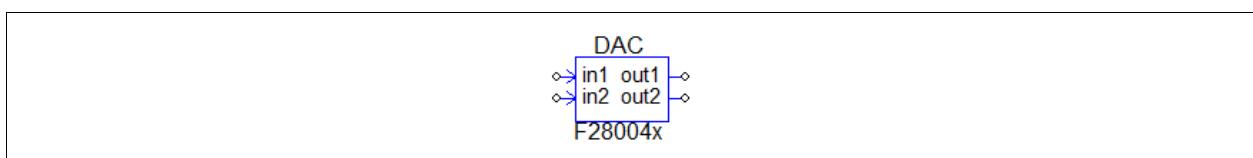
Use VDAC (the voltage reference is specified in Voltage Reference block):

$$DAC_Output = DAC_Input * VDAC / DAC_Range$$

11.12 D/A Converter

F28004x supports two 12-bit buffered D/A converter outputs.

Image:



Attributes:

Parameters	Description
Usage of DAC- x	The voltage reference used by DAC- x , where x is <i>A, B, or C</i> . It can be one of the followings: - <i>Do not use DAC-x</i> : DAC- x is not used. - <i>ADC Voltage Reference</i> : DAC- x is used in the system and ADC- x voltage reference is chose by DAC- x . - <i>VDAC Voltage Reference</i> : DAC- x is used in the system and VDAC voltage reference is chose by DAC- x .
DAC- x Input Range	The input range of DAC- x , where x is <i>A</i> or <i>B</i> .
DAC- x Output Gain	Specify the output gain of DAC- x . It can be 1 or 2 times. 2 times is allowed only if DAC-A uses ADC voltage reference and ADC voltage reference is set to internal +1.65V.
Use PWM Sync. for ADC- x	Specify if DAC- x is synchronized with a PWM block, where x is <i>A, B, or C</i> . The choices are <i>Do not use</i> or one of the PWM blocks in the pull-down list.
DAC- x Initial Value	Initial value of DAC- x , where x is <i>A, B, or C</i> .

All external high voltage reference (include VDAC) can be set in the range of 2.4V to 3.3V. The internal ADC voltage reference can be set to either 1.65V or 2.5V. DAC should use 2time gain if internal 1.65V is selected. All low voltage references are zero.

If DAC-B uses ADC voltage reference, and the ADC voltage reference is set to be external voltage reference, different CPU packages will have different voltage reference for DAC-B, as show in the table below:

	100-pin CPU	64-pin/56-pin CPU
DAC-A	VREFHIA/VREFLOA	VREFHIA/VREFLOA
DAC-B	VREFHIB/VREFLOB	VREFHIA/VREFLOA

If using VDAC as voltage reference, DACs and Comparator DACs can use VDAC as voltage reference; otherwise VDAC can not be selected by DACs and Comparator DACs. PSIM does not need ADC-B0 to connect a voltage reference in schematic, but in target system, ADC-B0 must connect a voltage reference.

The voltage reference of DACs can be VDAC or the voltage reference of ADC-A/B. If VDAC is chose, ADC-B0 must be set as VDAC. In Psim schematic, this should be only specified in Voltage Reference block, no need to link a voltage source to ADC-B0. But in target hardware, a voltage reference should be linked to ADC-B0. The output can be calculated as follows:

To use VDAC (the voltage reference is specified in Voltage Reference block):

$$DAC_Output = DAC_Input * VDAC / DAC_Range$$

To use ADC-A voltage reference (for DAC-A and DAC-B only):

$$DAC_Output = DAC_Input * Vref_ADC-A / DAC_Range$$

To use ADC-B voltage reference (for DAC-C only):

$$DAC_Output = DAC_Input * Vref_ADC-B / DAC_Range$$

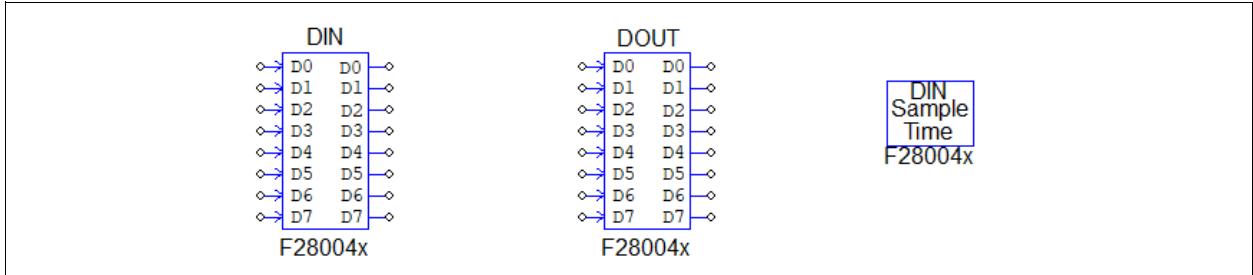
11.13 Digital Input, Output, and Sample Time

The digital input and output of F28004x can be any GPIO port from GPIO0 to GPIO59, or AIO port from AIO224 to AIO246. The relationship between AIO and ADC/comparator is as show in the table in section **11.9 A/D Converter**.

In SimCoder, the digital inputs and outputs are grouped in 8-channel blocks. Multiple 8-channel digital input/

output blocks can be used in the same schematic. All unused inputs should be connected to ground. The time of sampling period is specified at Digital Input Sample Time element.

Images:



Attributes for Digital Input:

Parameters	Description
Port Position for Input <i>i</i>	The port position of the Input <i>i</i> , where <i>i</i> is from 0 to 7. It can be any one of the GPIO ports.
Invert the Input <i>i</i>	Specify if the signal at Input <i>i</i> is inverted or not.
Use Pull-Up Resistor for Input <i>i</i>	Specify if the Input <i>i</i> has a pull-up resistor at the GPIO port.
Qualification for Input <i>i</i>	There are following qualification options: - Synchronize to CPU clock - 3 sampling period - 6 sampling period - No sync. or qualification

Attributes for Digital Output:

Parameters	Description
Port Position for Output <i>i</i>	The port position of the Output <i>i</i> , where <i>i</i> is from 0 to 7. It can be any one of the GPIO ports.
Port Mode for Output <i>i</i>	Port mode of the output <i>i</i> , where <i>i</i> ranges from 0 to 7. It can be any of the followings: - Normal output - Output with pull-up resistor - Open drain output

Note that each GPIO port can be used for one function only. If a GPIO port is used as a digital input port, it can not be used as a digital output or any other peripheral port. For example, if Port GPIO0 is assigned as digital input and it is also used as PWM1 output, an error will be reported.

Attributes for Digital Input Sample Time:

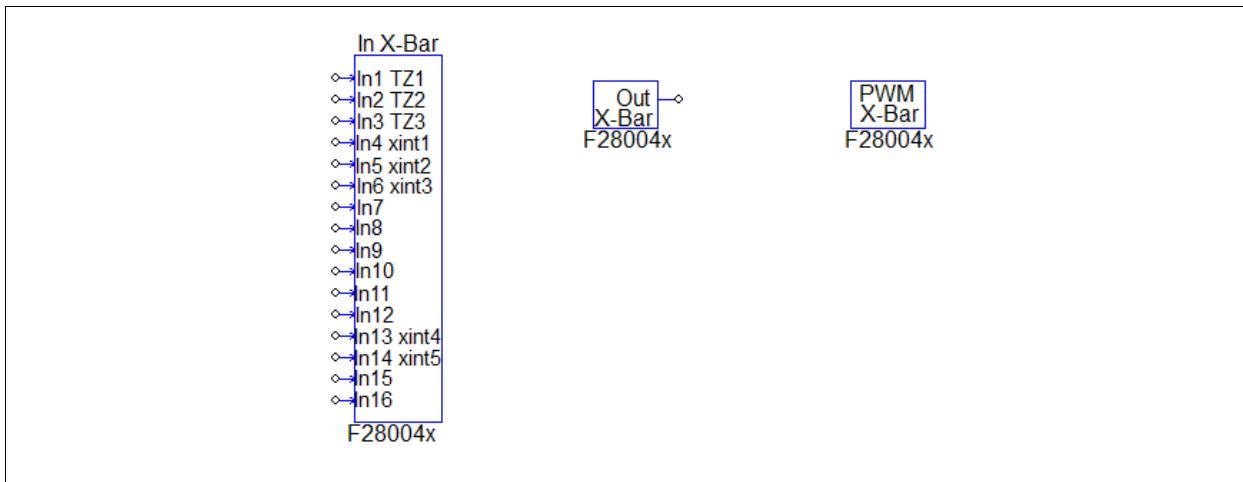
Parameters	Description
Set Gpio <i>x</i> Sample Time	Specify if setting sample time is used for GPIO group <i>x</i> . where $x = (8*i) - (8*i+7)$ with <i>i</i> ranges from 0 to 7.
Gpio <i>x</i> Sample Time (us)	The sample time for GPIO group <i>x</i> .
Set Aio <i>x</i> Sample Time	Specify if setting sample time is used for AIO group <i>x</i> , where $x = 200 + [(i*8) - (8*i+7)]$, and <i>i</i> ranges from 3 to 5
AIO <i>x</i> Sample Time (us)	The sample time for AIO group <i>x</i> .

11.14 Input X-BAR, Output X-BAR, and PWM X-BAR

The crossbars (referred to as X-BAR throughout this document) provide flexibility to connect device inputs, outputs, and internal resources in a variety of configurations. The F28004x device contains three X-BARs. Each of the X-BARs is named according to where they take signals:

- Input X-BAR: brings external signals “in” from a GPIO to the device
- Output X-BAR: takes internal signals “out” of the device to a GPIO.
- PWM X-BAR: takes signals and brings them to the PWM modules.

Images:



11.14.1 Input X-BAR

No more than one Input X-BAR is allowed in each F28004x system.

Attributes:

Parameters	Description
GPIO No. for Input <i>i</i>	Specify which GPIO/AIO port is used as input X-Bar signal <i>i</i>

Only one Input X-Bar block is allowed in each F28004x system, accepting 16 input X-Bar signals.

- Any GPIO/AIO port used as Input X-Bar signal can also be used as any other functionality. For example, GPIO0 is used as PWM1A, but it can also be used as Input X-Bar signal.
- INPUT1, INPUT2 and INPUT3 can be also used as TZ1, TZ2 and TZ3.
- INPUT4, INPUT5, INPUT6, INPUT13 and INPUT14 can be also used as XINT1, XINT2, XINT3, XINT4 and XINT5.

All input X-Bar signals can be also used as input capture signals.

11.14.2 Output X-BAR

There are 8 Output X-Bar signals, They are OUTPUT1 through OUTPUT8. These signals can be used to take the inner signals out to GPIO ports.

Attributes:

Parameters	Description
Output X-BAR No.	Specify Output X-Bar signal. It can be one of Output1 through Output8. Each output can be linked to one of multiple pre-decided GPIO ports.
Invert the Output	Specify if to invert the output.

Output Type	Specify GPIO output type. It can be one of the followings: - <i>Normal Output</i> : Normal output without pull-up resistor. - <i>Output with pull-up</i> : GPIO port links to an inner pull-up resistor. - <i>Open drain output</i> : GPIO port is an open drain output (SimCoder doesn't support this feature in simulation).
Signal Selection for MUX <i>i</i>	Specify the PWM X-Bar signal, as explained below.

SimCoder only supports the following signals as the source of the output X-Bar. Multiple input sources are allowed. The output is the OR of all selected sources.

MUX No.	Function 0	Function 1	Function 2	Function 3
0	CMPSS1.CTRIPOUTH	CMPSS1.CTRIPOUTH_OR_CTRIPOUTL	N/A	ECAP1OUT
1	CMPSS1.CTRIPOUTL	INPUTXBAR1	N/A	N/A
2	CMPSS2.CTRIPOUTH	CMPSS2.CTRIPOUTH_OR_CTRIPOUTL	N/A	ECAP2OUT
3	CMPSS2.CTRIPOUTL	INPUTXBAR2	N/A	N/A
4	CMPSS3.CTRIPOUTH	CMPSS3.CTRIPOUTH_OR_CTRIPOUTL	N/A	ECAP3OUT
5	CMPSS3.CTRIPOUTL	INPUTXBAR3	N/A	N/A
6	CMPSS4.CTRIPOUTH	CMPSS4.CTRIPOUTH_OR_CTRIPOUTL	N/A	ECAP4OUT
7	CMPSS4.CTRIPOUTL	INPUTXBAR4	N/A	N/A
8	CMPSS5.CTRIPOUTH	CMPSS5.CTRIPOUTH_OR_CTRIPOUTL	N/A	ECAP5OUT
9	CMPSS5.CTRIPOUTL	INPUTXBAR5	N/A	N/A
10	CMPSS6.CTRIPOUTH	CMPSS6.CTRIPOUTH_OR_CTRIPOUTL	N/A	ECAP6OUT
11	CMPSS6.CTRIPOUTL	INPUTXBAR6	N/A	N/A
12	CMPSS7.CTRIPOUTH	CMPSS7.CTRIPOUTH_OR_CTRIPOUTL	N/A	N/A
13	CMPSS7.CTRIPOUTL	N/A	N/A	N/A

11.14.3 PWM X-BAR

All F28004x's PWM DC trip-zone signals come from the PWM X-Bar,

Attributes:

Parameters	Description
PWM X-BAR No.	Specify PWM X-Bar signal. It can be one of the followings: <i>TRIP4</i> , <i>TRIP5</i> , and <i>TRIP7</i> through <i>TRIP12</i> . There is no <i>TRIP6</i> .
Invert the Output	Specify if to invert the output.
Signal Selection for MUX <i>i</i>	Specify the PWM X-Bar signal, as explained below.

There are 11 PWM X-Bar signals, They are TRIP1 - TRIP5, TRIP7 - TRIP12. There is no TRIP6 signal. These signals are used as DC trip-zone signals for PWM.

TRIP1 - TRIP3 are directly wired to TZ1 - TZ3 in Input X-Bar. The other signals need to set in the PWM X-Bar element. SimCoder only supports the following signals as PWM X-Bar input source. Multiple input sources are allowed. The output is the logic OR of all selected input sources.

MUX No.	Function 0	Function 1	Function 2	Function 3
0	CMPSS1.CTRIPH	CMPSS1.CTRIPH_OR_CTRIPL	N/A	ECAP1OUT

1	CMPSS1.CTRIPL	INPUTXBAR1	N/A	N/A
2	CMPSS2.CTRIPH	CMPSS2.CTRIPH_OR_CTRIPL	N/A	ECAP2OUT
3	CMPSS2.CTRIPL	INPUTXBAR2	N/A	N/A
4	CMPSS3.CTRIPH	CMPSS3.CTRIPH_OR_CTRIPL	N/A	ECAP3OUT
5	CMPSS3.CTRIPL	INPUTXBAR3	N/A	N/A
6	CMPSS4.CTRIPH	CMPSS4.CTRIPH_OR_CTRIPL	N/A	ECAP4OUT
7	CMPSS4.CTRIPL	INPUTXBAR4	N/A	N/A
8	CMPSS5.CTRIPH	CMPSS5.CTRIPH_OR_CTRIPL	N/A	ECAP5OUT
9	CMPSS5.CTRIPL	INPUTXBAR5	N/A	N/A
10	CMPSS6.CTRIPH	CMPSS6.CTRIPH_OR_CTRIPL	N/A	ECAP6OUT
11	CMPSS6.CTRIPL	INPUTXBAR6	N/A	N/A
12	CMPSS7.CTRIPH	CMPSS7.CTRIPH_OR_CTRIPL	N/A	N/A
13	CMPSS7.CTRIPL	N/A	N/A	N/A

11.15 Encoder, Encoder State, and Encoder Index/Strobe Position

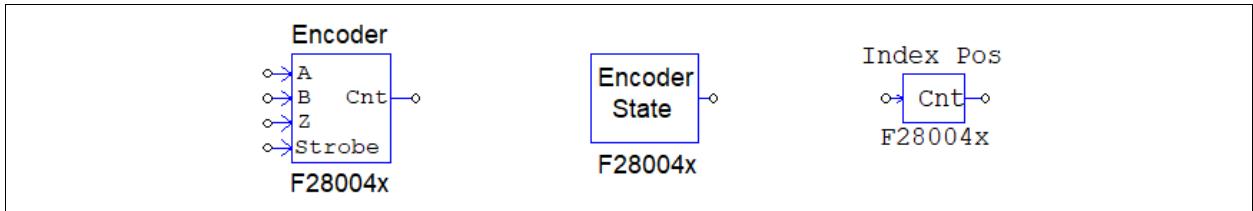
PSIM provides three blocks to execute encoder functions: **Encoder** block, **Encoder State** block, and **Encoder Index/Strobe Position** block.

F28004x supports two **Encoder** modules. Each Encoder module has several GPIO port options. The eQEP inputs include two pins for quadrature-clock mode or direction-count mode, an index (or Zero marker), and a strobe input. The output of the encoder element is the counter value. Also, hardware interrupt can be generated by the Z signal and the strobe signal.

The **Encoder State block** is used to indicate which input signal (either index signal or strobe signal) generates the interrupt. Also, hardware interrupt can be generated by the Z (index) signal and the strobe signal, and the output of the encoder state indicates which signal generates the interrupt. When the output is 0, the index signal generates the interrupt. When the output is 1, the strobe signal generates the interrupt.

The **Encoder Index/Strobe Position block** is used to latch the encoder's initial position. When the input of this block is 0, the encoder counter is set to 0. Encoder will start to act when the input changes to 1. Encoder will latch the counter value when it meet the index/strobe event. Use this block only when an encoder block in the schematic, and the encoder source and Z/Strobe signal are the same as specified in this block.

Images:



Attributes for Encoder:

Parameters	Description
Encoder Source	The source of the encoder. It can be any of the 2 encoders and with any of the GPIO port options.

Use Z Signal	Define the encoder to use the Z (or index) signal to initialize or latch the position counter on the occurrence of a desired event on the index pin. The options are: No: do not use Z signal Yes (rising edge): use Z signal and latch on rising edge. Yes (falling edge): use Z signal and latch on falling edge.
Use Strobe Signal	Define the encoder to use the strobe signal to initialize or latch the position counter on the occurrence of a desired event on the strobe pin. The options are: No: do not use strobe signal Yes (rising edge): use strobe signal and latch on rising edge. Yes (falling edge): use strobe signal and latch on falling edge.
Counting Direction	The counting direction - Forward: the encoder counts up. - Reverse: the encoder counts down.
Z Signal Polarity	Define the trigger polarity of Z signal. - Active High - Active Low.
Strobe Signal Polarity	Define the trigger polarity of strobe signal. - Active High - Active Low.
Encoder Resolution	The resolution of the external encoder hardware. If it is 0, the encoder counter will keep on counting and will not reset. If for example, the resolution is set to 4096, the counter will be reset to 0 after it reaches 4095.

Attributes for Encoder State:

Parameters	Description
Encoder Source	The source of the encoder. It can be any of the 2 encoders and with any of the GPIO port options.

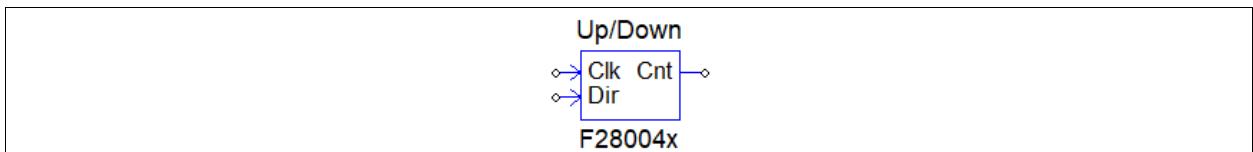
Attributes for Encoder Index/Strobe Position:

Parameters	Description
Encoder Source	The source of the encoder. It can be any of the 3 encoders and with any of the GPIO port options.
Latch Position	Specify the kept counter type, choose from the followings: - <i>IndexPos</i> , if the setting "Use Z Signal" is not "No" in Encoder - <i>StrobePos</i> , if the setting "Use Strobe Signal" is not "No" in Encoder
Type of Position	This can be chosen from the followings: - <i>The first latched position</i> , or - <i>The current latched position</i>

11.16 Up/Down Counter

F28004x supports one up/down counter for each of the two encoder modules.

Image:



Attributes:

Parameters	Description
Counter Source	The source of the counter. It can be any of the 2 encoders and with any of the GPIO port options listed in the pull-down menu.

There are two inputs for Up/Down Counter:

- Input "Clk" refers to the input clock signal.
- Input "Dir" refers defines the counting direction.

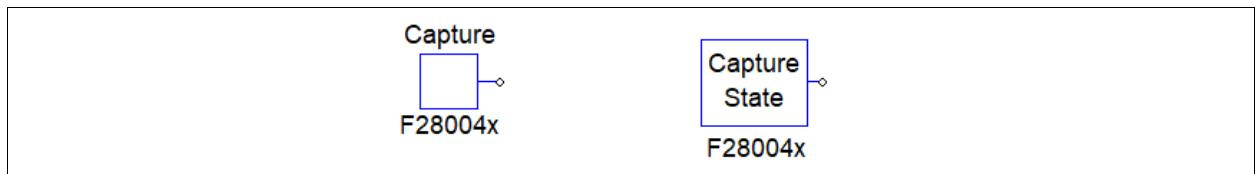
When the "Dir" input is 1, the counter counts forward (up), and when the input is 0, the counter counts backward (down).

The output of the up/down counter is the counter value.

Note that the up/down counter shares the same input GPIO ports with the encoder. Therefore, using both encoder and up/down counter on the same GPIO ports is not allowed.

11.17 Capture and Capture State

F28004x contains six enhanced capture module. PSIM use Capture block and Capture State block for capture function setup.

Image:**Attributes for Capture:**

Parameters	Description
Capture Source	Source of the capture. It may come from one of the seven captures.
Event Filter Prescale	Event filter prescale. The input signal is divided by the selected prescale.
Timer Mode	Capture counter timer mode. It can be one of the followings: <ul style="list-style-type: none">- <i>Absolute time</i>: The output is the absolute time in system clock- <i>Time difference (clock)</i>: The output is the difference between the current event and the previous event, in system clock.- <i>Time Difference (us)</i>: The output is the difference between the current event and the previous event, in us.

A capture can generate interrupt, and the interrupt trigger mode is defined in the interrupt block. If there is no the corresponding interrupt block in the schematic, Capture will catch rising edge only.

Capture input comes from Input X-Bar, so if a capture is used in the system, the corresponding input signal of Input X-Bar must be defined too.

Attributes for Capture State:

Parameters	Description
Capture Source	Source of the capture. It has only one source <i>Capture1</i> .

The Capture State block indicates the triggering state of the capture. Its output is either 1 or 0. 1 means the rising edge and 0 means the falling edge.

11.18 Serial Communication Interface (SCI)

F28004x provides the function for serial communication interface (SCI) functions.

PSIM utilizes the SCI data communication to run SimCoder's DSP Oscilloscope. This provides a convenient way to control and monitor the DSP code operation in real time. Through SCI, the data inside the DSP memory can be transferred to a computer through an external RS-232 cable and displayed on the PSIM's DSP Oscilloscope screen.

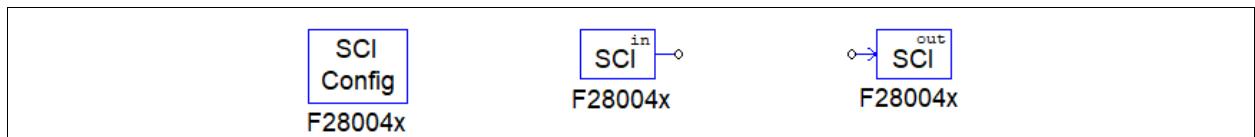
To monitor a variable inside DSP in real-time operation, an "SCI Output" block must be connected to that node on the schematic. Similarly, to be able to adjust a control variable in real-time, an "SCI Input" block must be assigned.

The values of all the global variables can be modified through the DSP Oscilloscope the same way as SCI inputs. For example, if the gain of a regulator is defined as a global variable, this gain can be modified through SCI while the code is running.

For more detailed descriptions on SCI and how to use the monitoring function, please refer to the document "*Tutorial - Using SCI for Real-Time Monitoring.pdf*".

Three SCI function blocks are provided in SimCoder: *SCI Configuration*, *SCI Input*, and *SCI Output*, as described below.

Images:



11.18.1 SCI Configuration

F28004x CPU supports two SCI ports: SCIA and SCIB. Each port has several options of GPIO pin combinations.

The SCI Configuration block defines the GPIO ports, the communication speed, the parity check type, and the data buffer size for SCI.

Attributes:

Parameters	Description
SCI RX Port	Select SCI Receiving port.
SCI TX Port	Select SCI transmission port. This port should be in the same SCI group with SCI receive port.
Speed (bps)	SCI communication speed, in bps (bits per second). A list of preset speeds is provided at 200000, 115200, 57600, 38400, 19200, or 9600 bps. Or one can specify any other speed manually.
Parity Check	The parity check setting for error check in communication. It can be either <i>None</i> , <i>Odd</i> , or <i>Even</i> .
Output Buffer Size	The buffer size that is allocated in the DSP RAM area for SCI data. Each buffer cell uses three 16-bit words. (that is, 6 bytes, or 48 bits, per data point).

Note that the buffer size should be properly selected. On one hand, a large buffer is preferred in order to collect more data points so that more variables can be monitored over a longer period of time. On the other hand, the internal DSP memory is limited, and the buffer should not be too large to interfere with the normal DSP operation.

11.18.2 SCI Input

The SCI Input block is used to define a variable in the DSP code that can be changed. The name of the SCI input variable will appear in the DSP Oscilloscope (under the **Utilities** menu), and the value can be changed at runtime via SCI.

Attributes:

Parameters	Description
Initial Value	The initial value of the SCI input variable.

The SCI input element provides a convenient way to change the values of variables such as a reference or controller parameters in real-time operation.

In PSIM schematic, the SCI Input behaves as a constant for simulation. The value is determined by the initial value in the parameter block. When code is generated, on the DSP oscilloscope provide by PSIM, the SCI inputs are listed as input variables. The user may change the values while the code is running in real time.

11.18.3 SCI Output

The SCI Output block is used to define a variable for display. When a SCI output block is connected to a node, the name of the SCI output block will appear in the DSP Oscilloscope (under the **Utilities** menu), and data of this variable can be transmitted from DSP to the computer via SCI at runtime, and the waveform can be displayed in the DSP Oscilloscope.

The SCI output block provides a convenient way to monitor DSP waveforms.

Attributes:

Parameters	Description
Data Point Step	It defines how frequent data is collected. If the Data Point Step is 1, every data point is collected and transmitted. If the Data Point Step is 10, for example, only one point out of every 10 points is collected and transmitted.

Note that if the Data Point Step is too small, there may be too many data points and it may not be possible to transmit them all. In this case, some data points will be discarded during the data transmission.

Also, the Data Point Step parameter is used only when the DSP Oscilloscope is in the *continuous* mode. When it is in the *snapshot* node, this parameter is ignored and every point is collected and transmitted.

In simulation, the SCI output behaves as a voltage probe.

11.19 Serial Peripheral Interface (SPI)

F28004x provides the functions for serial peripheral interface (SPI). The SPI blocks in the TI F28004x Target library provide convenience to implement the function for communication with external SPI devices (such as external A/D and D/A converters).

Writing code manually for SPI devices is often a time-consuming and non-trivial task. With the capability to support SPI, PSIM greatly simplifies and speeds up the coding and hardware implementation process.

SimCoder only supports SPI used as a master, and other SPI devices connected to DSP as slave.

In a system using SPI functions, the SPI configuration element must be present in the schematic. For each SPI Device in the system, user puts a SPI device element in the schematic. Each SPI Device can be used either as an input SPI device or as an output SPI device. If a SPI element has both input and output channels, user should put 2 SPI devices in the schematic, one as an input SPI device and another as an output device, both of them share the same chip selection pins.

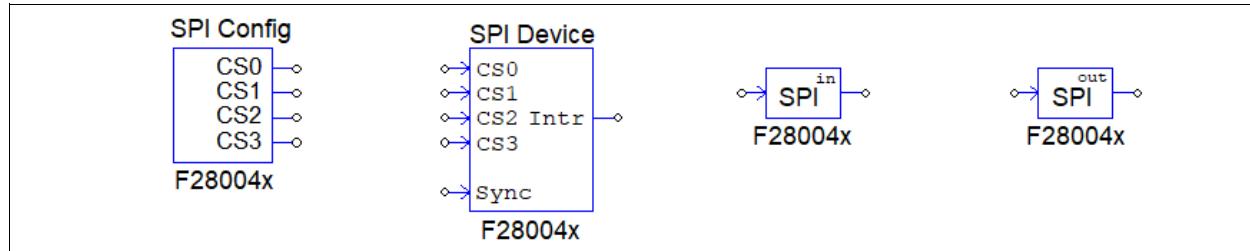
In a PSIM schematic, a system can use up to 16 SPI devices. User must design a proper circuit to generate chip select signals for each SPI devices. If the generated program needs to issue multiple commands in one cycle, user must check the time gap between any two cycle command series. If this time gap is too short, SPI commands may get lost.

For more detailed descriptions on how to use SPI blocks, please refer to the document "*Tutorial - Using SPI for*

Real-Time Monitoring.pdf".

Four SPI function blocks are provided in SimCoder: *SPI Configuration*, *SPI Device*, *SPI Input*, and *SPI Output*, as described below.

Images:



11.19.1 SPI Configuration

The SPI Configuration block defines the SPI port, the chip selection pins, and the SPI buffer size. It must be present in a schematic where SPI is used, and this block must be in the main schematic.

Attributes:

Parameters	Description
SPI SIMO Port	Specify SIMO(SPI data output) port.
SPI SOMI Port	Specify SOMI(SPI data input) port. This port should have the same SPI source with SIMO port.
SPI CLK Port	Specify CLK(SPI clock) port. This port should have the same SPI source with SIMO port.
SPI STE Port	Specify STE(SPI STE) port. This port should have the same SPI source with SIMO port.
Chip Select Pin <i>i</i>	The GPIO port of the chip select pin. PSIM supports four GPIO pins for chip select, as defined by Chip Select Pin0 to Pin3. These GPIO ports and the SPI slave transmit-enable pin SPISTE are used to generate the chip select signal.
SPI Buffer Size	The buffer size of the SPI commands. Each memory cell of the buffer saves the index of a SPI command. Normally, one can specify the buffer size as 1 plus the number of SPI commands (i.e. Start Conversion Command, Receiving Data Command, Sending Data Command, and Sync. Command) in all SPI Input/Output elements.

11.19.2 SPI Device

The SPI Device block defines the information of the corresponding SPI hardware device. The number of SPI Device blocks in the schematic must be the same as the number of SPI hardware devices.

Attributes:

Parameters	Description
Chip Select Pins	The state of the chip select pins corresponding to the SPI device. When the chip select pins are at this state, this SPI device is selected.
Communication Speed (MHz)	SPI communication speed, in MHz.

Clock Type	SPI clock type, as determined by the SPI hardware device. It can be one of the following: <ul style="list-style-type: none"> - <i>Rising edge without delay</i>: The clock is normally low, and data is latched at the clock rising edge. - <i>Rising edge with delay</i>: The clock is normally low, and data is latched at the clock rising edge with delay. - <i>Falling edge without delay</i>: The clock is normally high, and data is latched at the clock falling edge. - <i>Falling edge with delay</i>: The clock is normally high, and data is latched at the clock falling edge with delay.
Command Word Length	Word length, or the length of the significant bits, of SPI communication commands. It can be from 1 to 16 bits.
Sync. Active Mode	The triggering mode of the synchronization signal of the SPI device. It can be either <i>Rising edge</i> or <i>Falling edge</i> .
SPI Initial Command	The SPI command that initializes the SPI device.
Hardware Interrupt Mode	Specify the type of the interrupt signal that the SPI device generates. This is valid only when the SPI device's interrupt output node is connected to the input of a digital output element. It can be one of the following: <ul style="list-style-type: none"> - <i>No hardware interrupt</i> - <i>Rising edge</i> - <i>Falling edge</i>
Conversion Type	Specify the conversion type of this SPI device. It can be one of following: <ul style="list-style-type: none"> Conversion in series: All SPI conversions are in series. Conversion in parallel: All SPI conversions are in parallel.
Command Gaps (ns)	The gap between two SPI commands, in nanosecond. This can be also used as ADC conversion time.
Conversion Sequence	Arrange the names of the SPI input elements, separated by comma, in the conversion sequence. Note that this parameter is valid only when the SPI device generates multiple interrupts in series

In a schematic, the chip select pins of all the SPI devices are connected to the chip select pins of the SPI Configuration block, without defining how the chip select logic is implemented. In the actual hardware, however, one would need to implement the corresponding chip select logic accordingly.

A SPI command consists of a series of 16-bit numbers separated by comma. In the 16-bit number, only the lower bits are the significant bits used by the command. For example, if the Command Word Length is 8, Bits 0 to 7 are the command, and Bits 8 to 15 are not used.

A SPI device can be either an input device or an output device. For example, an external A/D converter is an input device. Usually DSP will send one or multiple A/D conversion commands to the device, and then set the synchronization signal to start the conversion. The synchronization signal is reset at the next command of the same device.

A SPI input device using the synchronization signal usually needs an interrupt pin to trigger DSP to enter the interrupt service routine.

On the other hand, an external D/A converter is an output device. Usually DSP sends one or multiple D/A conversion commands to the device, and then sets the synchronization signal to start the conversion. The synchronization signal is reset at the next command of the same device.

11.19.3 SPI Input

A SPI input device may have multiple input channels. The SPI Input block is used to define the properties of an input channel for SPI communication, and one SPI Input block corresponds to one input channel.

Attributes:

Parameters	Description
Device Name	Name of the SPI input device.
Start Conversion Command	Command to start conversion, in hex numbers, separated by comma (for example, 0x23,0x43,0x00).
Receiving Data Command	Command to receive data, in hex numbers, separated by comma (for example, 0x23,0x43,0x00).
Data Bit Position	Define where the data bits are in the receiving data string. The format is: $\text{ElementName} = \{Xn[\text{MSB...LSB}]\}$ where <ul style="list-style-type: none"> - ElementName is the name of the SPI input device. If it is the current SPI input device, use y instead. - $\{\}$ means that the item in the bracket repeats multiple times. - Xn is the n_{th} word received from the SPI input device, and n start from 0. - MSB...LSB defines the position of the significant bits in the word.
Input Range	Specify the parameter V_{max} that defines the input range. This parameter is valid only when the SPI device is an A/D converter. If the device conversion mode is DC, the input ranges from 0 to V_{max} . If the device conversion mode is AC, the input ranges from $-V_{max}/2$ to $V_{max}/2$.
Scale Factor	Output scale factor K_{scale} . If the scale factor is 0, the SPI device is not an A/D converter, and the result will be exactly the same as what DSP receives from SPI communication. Otherwise, the SPI device is an A/D converter, and the result is scaled based on this factor and the A/D conversion mode.
ADC Mode	The A/D conversion mode of the device. It can be either DC or AC. Note that this parameter is valid only when the device is an A/D converter.
Initial Value	The initial value of the input.

The formula for the *Data Bit Position* defines the data length of a SPI input device. For example, $y=x1[3..0]x2[7..0]$, means that the data length is 12, and the result is the lower 4 bits of the 2nd word and the lower 8 bits of the 3rd word. If the received data string is 0x12,0x78,0xAF, then the result is 0x8AF.

If the scale factor is not 0, the output will be scaled based on the following:

In the DC conversion mode:

- In simulation:
$$\text{Output} = \text{Input} \cdot K_{scale}$$
- In hardware:
$$\text{Output} = \frac{\text{Result} \cdot V_{max} \cdot K_{scale}}{2^{\text{Data_Length}}}$$

In the AC conversion mode:

- In simulation:
$$\text{Output} = \text{Input} \cdot K_{scale}$$
- In hardware:
$$\text{Output} = \frac{(\text{Result} - 2^{\text{Data_Length}-1}) \cdot V_{max} \cdot K_{scale}}{2^{\text{Data_Length}-1}}$$

The parameter *Data Length* is calculated from the Data Bit Position formula.

11.19.4 SPI Output

A SPI output device may have multiple output channels. The SPI Output block is used to define the properties of an output channel for SPI communication, and one SPI Output block corresponds to one output channel.

Attributes:

Parameters	Description
Device Name	Name of the SPI output device.
Scale Factor	Output scale factor K_{scale} . If the scale factor is 0, the SPI device is not a D/A converter, and the result will be exactly the same as what DSP receives from SPI communication. Otherwise, the SPI device is an D/A converter, and the result is scaled based on this factor and the D/A conversion mode.
Output Range	Specify the parameter V_{max} that defines the output range. This parameter is valid only when the SPI device is an D/A converter. If the device conversion mode is DC, the input ranges from 0 to V_{max} . If the device conversion mode is AC, the input ranges from $-V_{max}/2$ to $V_{max}/2$.
DAC Mode	The D/A conversion mode of the device. It can be either <i>DC</i> or <i>AC</i> . Note that this parameter is valid only when the device is a D/A converter.
Sending Data Command	Command to send the output data, in hex numbers, separated by comma (for example, 0x23,0x43,0x00).
Data Bit Position	Define where the data bits are in the sending data string. The format is: $ElementName = \{Xn[MSB..LSB]\}$ where - $ElementName$ is the name of the SPI output device. If it is the current SPI output device, use y instead. - $\{\}$ means that the item in the bracket repeats multiple times. - Xn is the n_{th} word sent to the SPI output device, and n start from 0. - $MSB..LSB$ defines the position of the significant bits in the word.
Sync. Command	The command to synchronize output channels of the SPI output device, in hex numbers, separated by comma (for example, 0x23,0x43,0x00). This command is used when the SPI output device does not have the synchronization signal

The formula for the *Data Bit Position* defines the data length of a SPI output device. For example, $y=x1[3..0]x2[7..0]$, means that the data length is 12, and the data is the lower 4 bits of the 2nd word and the lower 8 bits of the 3rd word. If the sending data string is 0x12,0x78,0xAF, then the data is 0x8AF.

If the scale factor is not 0, the output will be scaled based on the following:

In the DC conversion mode:

$$\text{- In simulation: } Output = Input \cdot K_{scale}$$

$$\text{- In hardware: } Output = \frac{Result \cdot K_{scale} \cdot 2^{Data_Length}}{V_{max}}$$

In the AC conversion mode:

$$\text{- In simulation: } Output = Input \cdot K_{scale}$$

$$\text{- In hardware: } Output = 2^{Data_Length} + \frac{Result \cdot K_{scale} \cdot 2^{Data_Length-1}}{V_{max}}$$

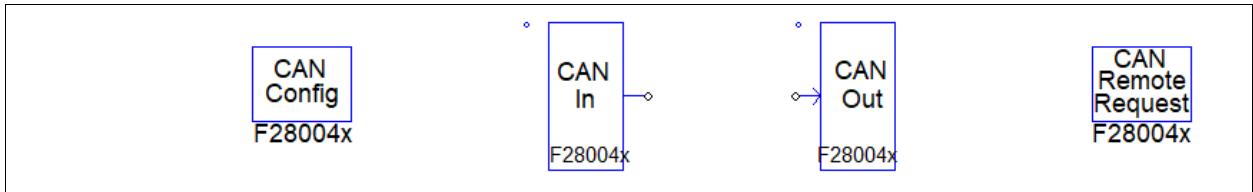
The parameter *Data_Length* is calculated from the Data Bit Position formula.

11.20 Controller Area Network (CAN) Bus

F28004x DSP has 2 CAN sources: DCAN-A and DCAN-B, and each has multiple choices for transmit pin and receive pin.

Four function blocks are provided in SimCoder: *CAN Configuration*, *CAN Input*, *CAN Output*, and *CAN Remote Request*, as described below.

Images:



11.20.1 CAN Configuration

The CAN Configuration block defines the CAN bus source, data byte order, and other settings. Attributes:

Parameters	Description
CAN RX Port	CAN receive port.
CAN TX Port	CAN transmit port. Both CAN RX/TX ports should be selected to either CAN-A or CAN-B.
CAN Speed	Communication speed, in Hz. It can be set to one of the following preset values: 125kHz, 250kHz, 500kHz, and 1MHz One can also set the speed manually by typing the text in the parameter field. Note that the speed should not exceed 1MHz.
Data Byte Order	The order of the data bytes. It can be one of the following: - <i>Least significant byte 1st</i> : the least significant byte is placed first - <i>Most significant byte 1st</i> : the most significant byte is placed first.
Number of Input Mailboxes	Number of input mailboxes.
Checking Receive Mail Lost	If it is set to <i>Enable</i> , the error report function " <code>_ProcCanAErrReport(nErr)</code> " (for CAN A) will be called if the received mail is lost. This function will return the value of the error status nErr from the CAN register CANGIF0. If it is set to <i>Disable</i> , the error report function will not be called.
Checking Bus Off	If it is set to <i>Enable</i> , the error report function " <code>_ProcCanAErrReport(nErr)</code> " (for CAN A) will be called if the bus is in the off state. This function will return the value of the error status nErr from the CAN register CANGIF0. If it is set to <i>Disable</i> , the error report function will not be called.
Error Check Mode	The error check mode can be either <i>Passive</i> or <i>Active</i> . If it is in the error-passive mode, an interrupt will be generated when the error count reaches 128. If it is in the error-active mode, an interrupt will be generated every time.

The variable nErr in the callback function "`_ProcCanAErrReport(nErr)`" (for DCAN A) is a 32-bit integer. It obtains its value from the Global Interrupt Flag Register CANGIF0. After returning from the function "`_ProcCanAErrReport(nErr)`", the register CANGIF0 will be cleared.

Also, if you wish to take actions on a specific error, you can add your own code within the "`_ProcCanAErrReport(nErr)`" function.

11.20.2 CAN Input

A CAN Input block receives CAN messages from a CAN bus.

Attributes:

Parameters	Description
Number of Inputs	Number of CAN inputs. It can have up to 8 inputs.
CAN Source	The source of CAN input, either CAN A or CAN B
Use Extension ID	If this is set to <i>Yes</i> , the ID of a message is a 29-bit integer. If this is set to <i>No</i> , the ID of a message is a 11-bit integer.
Message ID	The ID of a message. It is an integer, for example, 0x23.
Local Mask	The mask for the message. It is an integer. If the bits of the message ID matches the bits of the mask, the message will be received. Otherwise, it will be ignored. For example, if the mask is 0x380 and the message ID is 0x389, this message will be received. But if the message ID is 0x480, the message will be ignored.
Overwrite Protection Flag	It can be set to <i>Allow overwrite</i> or <i>Do not allow overwrite</i> . Assume a mailbox is configured to accept a message, and there is a new message coming from the CAN bus, while the old message is still in the mailbox and has not been processed yet. If the flag is set to "Allow overwrite", the new message will be accepted, and the old message will be overwritten. If the flag is set to "Do not allow overwrite", the new message will not be accepted, and the old one will be kept.
Receive Message Rate	The number of messages received by the input block in a specific period of time. For example, there are two input blocks, with the first input block receiving 20 messages per second, and the second input block receiving 30 messages per second. The parameter "Receive Message Rate" will be set to 20 for the first block, and 30 for the second block.
Input i Gain	The gain to the i_{th} input where i can be 1 to 8. The output is the input multiplied by the gain.
Input i Data Start Position	A message can have up to 8 data points. A data point can have 1 bit up to 32 bits. This defines the start position of the current data point in the message.
Input i Data End Position	This defines the end position of the current data point in the message.
Input i Data Type	The data type can be either <i>Float</i> , <i>Integer</i> , or <i>IQ1</i> to <i>IQ30</i> .
Input i Default Data	The initial value of the SCI input variable.

11.20.3 CAN Output

A CAN Output block transmits CAN messages to a CAN bus.

Attributes:

Parameters	Description
Number of Outputs	Number of CAN outputs. It can have up to 8 outputs.
CAN Source	The CAN source can be either CAN A or CAN B.
Use Extension ID	If this is set to <i>Yes</i> , the ID of a message is a 29-bit integer. If this is set to <i>No</i> , the ID of a message is a 11-bit integer.
Message ID	The ID of a message. It is an integer, for example, 0x23.

Trigger Type	<p>The trigger type can be one of the following:</p> <ul style="list-style-type: none"> - <i>No trigger</i>: No triggering - <i>Rising edge</i>: Triggering occurs at the rising edge of the trigger source. - <i>Falling edge</i>: Triggering occurs at the falling edge of the trigger source. - <i>Rising/falling edge</i>: Triggering occurs at both the rising edge and the falling edge of the trigger source. <p>A rising/falling edge is considered to have occurred if the difference between the current value of the trigger source and the value at the last triggering instant is equal to or greater than 1.</p>
Trigger Source	<p>A trigger source can be either a constant or a global variable depending on the Trigger Type.</p> <p>If the Trigger Type is set to "No trigger", the trigger source defines the counter limit. For example, if the trigger source is a constant or a global value, and the value is 5, it means that triggering will occur once out of every 5 times. In another word, the data will be sent out once per every 5 cycles.</p> <p>If the Trigger Type is set to edge trigger, the trigger source can only be a global variable. Triggering will occur when the global variable has the rising or falling edge or both depending on the Trigger Type.</p>
Output <i>i</i> Gain	The gain to the i_{th} output where i can be 1 to 8. The output is the output multiplied by the gain.
Output <i>i</i> Data Start Position	A message can have up to 8 data points. A data point can have 1 bit up to 32 bits. This defines the start position of the current data point in the message.
Output <i>i</i> Data End Position	This defines the end position of the current data point in the message.
Output <i>i</i> Data Type	The data type can be either <i>Float</i> , <i>Integer</i> , or <i>IQ1</i> to <i>IQ30</i> .
Output <i>i</i> Default Data	The initial value of the SCI output variable.

11.20.4 CAN Remote Request

The CAN Remote Request block sends a remote request to CAN bus, and receives the message from the remote frame with the same message ID on CAN bus.

Attributes:

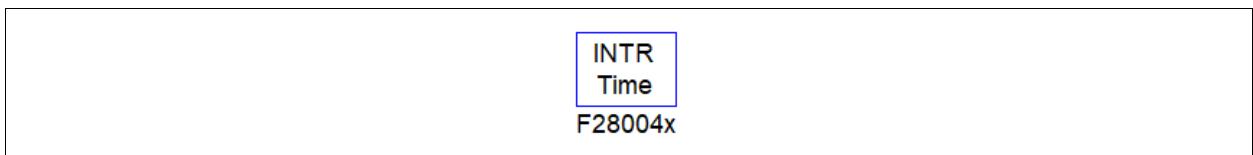
Parameters	Description
CAN Source	The source of CAN input, either CAN A or CAN B
Use Extension ID	If this is set to <i>Yes</i> , the ID of a message is a 29-bit integer. If this is set to <i>No</i> , the ID of a message is a 11-bit integer.
Message ID	The ID of a message. It is an integer, for example, 0x23.
Trigger Type	<p>The trigger type can be one of the following:</p> <ul style="list-style-type: none"> - <i>No trigger</i>: No triggering. - <i>Rising edge</i>: Triggering occurs at the rising edge of the trigger source. - <i>Falling edge</i>: Triggering occurs at the falling edge of the trigger source. - <i>Rising/falling edge</i>: Triggering occurs at both the rising edge and the falling edge of the trigger source. A rising/falling edge is considered to have occurred if the difference between the current value of the trigger source and the value at the last triggering instant is equal to or greater than 1.

Trigger Source	A trigger source can be either a constant or a global variable depending on the Trigger Type. If the Trigger Type is set to "No trigger", the trigger source defines the counter limit. For example, if the trigger source is a constant or a global value, and the value is 5, it means that triggering will occur once out of every 5 times. In another word, the data will be sent out once per every 5 cycles. If the Trigger Type is set to edge trigger, the trigger source can only be a global variable. Triggering will occur when the global variable has the rising or falling edge or both depending on the Trigger Type.
Frequency	The sampling frequency.

11.21 Interrupt Time

The interrupt time block is used to measure the time interval of an interrupt service routine.

Image:



Attributes:

Parameters	Description
Time Output Method	Define how interrupt time is measured. It can be one of the following: <ul style="list-style-type: none"> - <i>SCI (time used)</i>: Using SCI. Time used by the interrupt service routine, in DSP clock count, is measured and is sent out via SCI output. - <i>SCI (time remaining)</i>: Using SCI. Time remaining in the interrupt service routine, in DSP clock count, is measured and is send out via SCI output. The time remaining is defined as the time from the end of the current interrupt to the beginning of the next interrupt. - <i>GPIOi</i>, or <i>AIOi</i>: Using a GPIO or an AIO port. A pulse is generated at the specified GPIO port. The pulse is set to high when entering the interrupt, and set to low when exiting the interrupt. An oscilloscope can be used to measure the width of the pulse.
Sampling Frequency	Sampling frequency of the interrupt service routine, in Hz.

When SCI is used, the value is the count of the DSP clock. For example, if the value is 7500, for a 90-MHz DSP clock, the interrupt time will be:

$$7500 / 90M = 83.33\mu s$$

11.22 Project Settings and Memory Allocation

When generating the code for the TI F28004x Hardware Target, SimCoder also creates the complete project files for the TI Code Composer Studio (CCS) development environment, so that the code can be compiled, linked, and uploaded to the DSP.

At the present, ProjectSpec is supported for F28004x code. This is a new CCS project format and it is supported since CCS 5.3. Assuming that the PSIM schematic file is "test.sch", after the code generation, a sub-folder called "test (C code)" will be generated in the directory of the schematic file, and sub-folder will contain the following files:

- test.c Generated C code
- PS_bios.h: Header file for the SimCoder F28004x library
- passwords.asm: File for specifying the DSP code password

- test.projectspec: Project file for Code Composer Studio
- F28004x_Headers_NonBIOS.cmd: Peripheral register linker command file
- F28004x_FLASH_Lnk.cmd: Flash memory linker command file
- F28004x_FLASH_RAM_Lnk.cmd: Flash RAM memory linker command file
- F28004x_RAM_Lnk.cmd: RAM memory linker command file
- PsBiosRamF004xFLOAT.lib: PsBios library for ram code.
- PsBiosRamF004xFLOAT.lib: Psbios library for Flash code

Note: The names of the link command files are assigned with the real target hardware the schematic used. For example, if the target hardware is F280049, the file names will be F280049FLASHLink.cmd, F280049FlashRamLink.cmd, and F280049RamLink.cmd accordingly. These files will be copied automatically to the project folder when the code is generated.

Each time code generation is performed, the .c file and .projectspec file (in this example, "test.c" and "test.projectspec") will be created. If you have made changes manually to these two files, be sure to copy the changed files to a different location. Otherwise the changes will be overwritten when code generation is performed next time

Project Setting:

To load .projectspec project, please use CCS 5.3 or latter CCS versions. In CCS main menu, choose **Project/ Import CCS project...**, then browse the project folder in the dialog. CCS will automatically load the project in the folder.

In the Code Composer Studio project file, the following settings are provided:

- *RAM Debug*: To compile the code in the debug mode and run it in the RAM memory
- *RAM Release*: To compile the code in the release mode and run it in the RAM memory
- *Flash Release*: To compile the code in the release mode and run it in the flash memory
- *Flash RAM Release*: To compile the code in the release mode and run it in the RAM memory

When *RAM Debug* or *RAM Release* is selected, CCS uses the linker command file **F28004x_RAM_Lnk.cmd** to allocate the program and data space.

When *Flash Release* is selected, CCS uses the linker command file **F28004x_FLASH_Lnk.cmd** to allocate the program and data space.

When *Flash RAM Release* is selected, CCS uses the linker command file **F28004x_FLASH_RAM_Lnk.cmd** to allocate the program and data space. The memory allocation is the same as in *RAM Release*.

The code compiled in the release mode is faster than the code in the debug mode. Also, the code in *RAM Release* or *Flash RAM Release* is the fastest. The code in *RAM Debug* is slower, and the code in *Flash Release* is the slowest. In a development, normally one would start with *RAM Debug* for easy debugging. Then switch to *RAM Release* and consequently to *Flash Release* or *Flash RAM Release*.

PE-Expert4 Hardware Target

12.1 Overview

With the PE-Expert4 Hardware Target, SimCoder can generate the code for the PE-Expert4 DSP development platform from Myway with the following boards:

- DSP Board: MWPE4-C6657, the core DSP board
- PEV Board: MWPE4-PEV, the PWM generation board
- PSPWM Board: Extension boards including MWPE4-FPGA24 and MWPE4-FPGA6
- ADC Board: MWPE4-ADC, the analog signal input extension board

When generating the code for a system that has multiple sampling rates, SimCoder will use the PWM generator interrupts for the PWM sampling rates. It will then first use the Timer 0 interrupt, and then Timer 1 interrupt if needed, for other sampling rates in the control system. If there are more than three sampling rates in the control system, the corresponding interrupt routines will be handled in the main program by software.

In the PE-Expert4 system, digital input, encoder, and capture can also generate hardware interrupts. An interrupt must be associated with an interrupt service routine (a subcircuit that represents the interrupt service routine) through the interrupt block as described in Section 5.4 of this Manual. In PE-Expert4, since interrupts generated by digital input, encoder, and capture are handled by the same interrupt service routine, all the interrupt blocks must connect to the same subcircuit block.

The hardware functions and elements of the PE-Expert4 Hardware Target are described in the sections below.

12.2 DSP Board

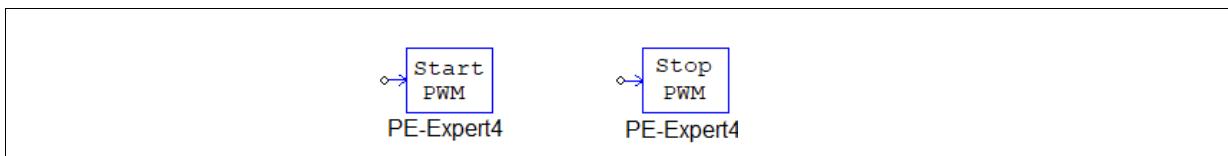
The DSP Board in the PE-Expert4 system is the core board. It contains the following functions and hardware elements:

- Start/ Stop PWM
- LED Output
- Timer Interrupt Priority

12.2.1 Start and Stop PWM

These block start or stops the PWM function for PE-Expert4 target.

Images:



Attributes:

Parameters	Description
Board No.	The number of the PEV board that contains the PWM generator to be started
Board Type	The board type of Expert4 board, can be either the PEV board or the PSPWM board.

To start PWM, apply a high logic signal (1) to the input of the "Start PWM" element. To stop PWM, apply a high logic signal (1) to the input of the "Stop PWM" element.

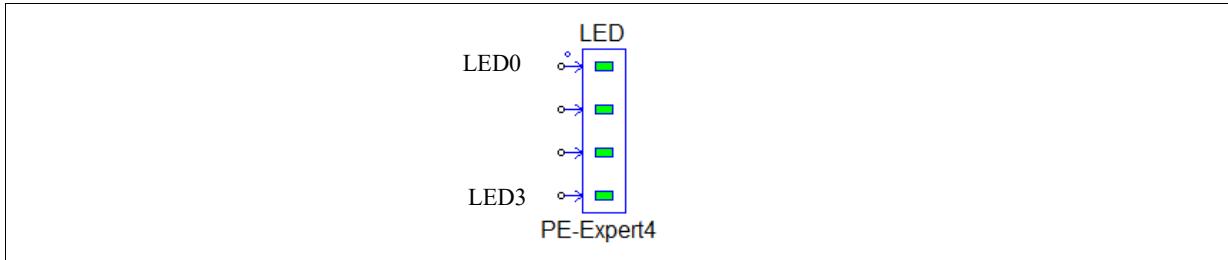
12.2.2 LED Output

The LED output element is available for the DSP Board MWPE4-C6657.

There are four light-emitting diodes (LED) on the DSP Board: LED0, LED1, LED2, and LED3. In the image, the input with the dot corresponds to the LED0.

When the input level of a LED is higher than 0.3, the LED will be on. Otherwise, it will be off.

Image:

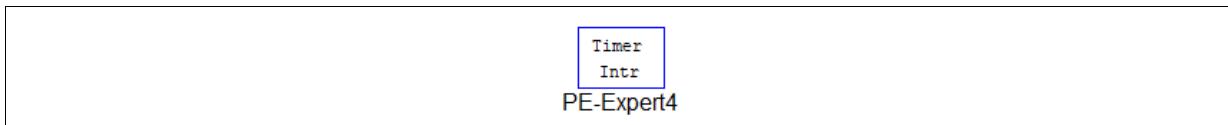


Note that this element is for hardware implementation only, and it will be ignored in the simulation. To display the LED value in the simulation, connect a voltage probe to the input node.

12.2.3 Timer Interrupt Priority

This element sets the timer interrupt priority.

Image:



Attributes:

Parameters	Description
Interrupt Priority of Timer0	Specify interrupt priority of Timer0. Options are from 4 (highest) to 13 (lowest).
Interrupt Priority of Timer1	Specify interrupt priority of Timer1.

This element is optional. The priority can be set from 4 (highest) to 13 (lowest). If no Timer Interrupt Priority element is in the schematic, SimCoder will set interrupt priority 10 to timer0, interrupt priority 11 to timer1.

12.3 PEV Board

The PEV board contains the following functions and hardware elements:

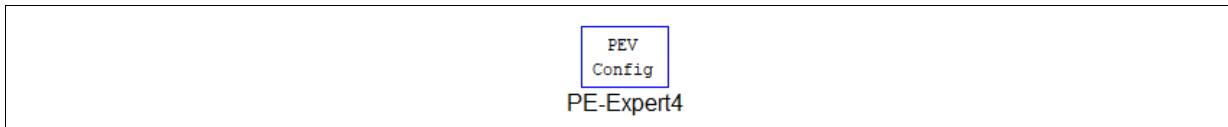
- PWM generators and Start/Stop PWM functions
- A/D converter
- Digital input/Capture/Counter
- Digital output
- Encoder
- PWV Board Configuration

Please note that hardware input/output elements, including PWM generators, A/D converter, digital input/output, up/down counter, encoder, and capture, can not be placed inside a subcircuit. They must be in the top-level main circuit only. The Start/Stop PWM function elements, however, can be placed in either the main circuit or subcircuits.

12.3.1 PEV Board Configuration

This element defines the PEV board configuration. If a PSIM circuit users any of the PEV board element, this configuration block must be used.

Image:



Attributes:

Parameters	Description
Board No.	The board number (0 to4) of the PEV board that contains these elements.
Interrupt Source of inti	Select the interrupt source that uses inti in the current system. The options are: ADC, PWM, Encoder, Capture, and Digital Input.
Interrupt Priority of inti	Select the interrupt priority that uses inti in the current system. The options are from 4 (highest) to 13 (lowest).

This element specifies all interrupt signals used by interrupt sources, for each interrupt signal, one can specify its interrupt priority, too.

For the interrupt sources of ADC/PWM combination, if PWM does not trigger ADC, the interrupt source should be set as *PWM*. If PWM triggers ADC, the interrupt source should be set as *ADC*.

12.3.2 PWM Generators

There are three elements for PWM generation:

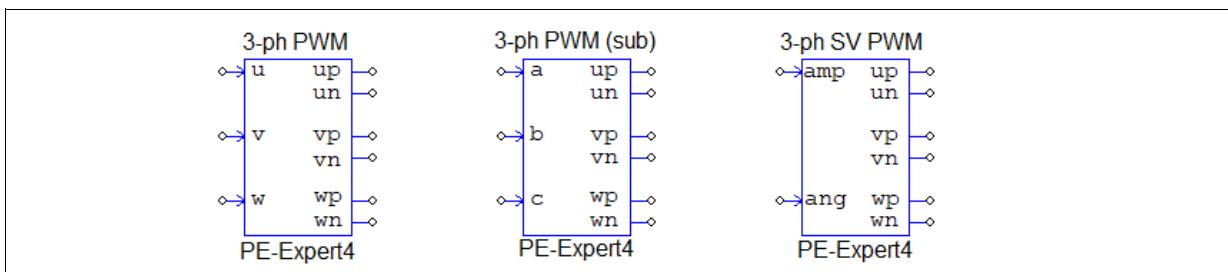
3-ph PWM generator,

3-ph PWM (sub) generator,

3-ph Space Vector PWM generator,

The **PWM (sub)** element consists of the PWM function and the input-scaling-offset circuit.

Images:



Attributes:

Parameters	Description
Board No.	The board number of the PEV board that contains the PWM generator.
Scaling Factor	Scaling Factor is an integer that ranges from 1 to 1000, specifies the number of PWM period between two PWM-caused-interrupts.

Interrupt Mode	The interrupt mode can be set to one of the following: <ul style="list-style-type: none"> - <i>Interrupt at peak</i>: Interrupt occurs at the peak of the carrier wave - <i>Interrupt at valley</i>: Interrupt occurs at the valley of the carrier wave - <i>Interrupt at peak/valley</i>: Interrupt occurs at both the peak and valley of the carrier wave When the interrupt mode is set to both peak and valley, the interrupt occurs at two times the rate of the carrier wave. In another word, if the carrier frequency is 10kHz, the interrupt rate will be 20kHz.
Interrupt Position	The interrupt position ranges from -0.5 to +0.5. If the value is positive, it is after the beginning of the PWM cycle. If the value is negative, it is before the beginning of the PWM cycle. If the A/D converter is used and its conversion mode is set to 'PWM Triggered', this parameter specifies the position that PWM triggers ADC for conversion. If PWM does not trigger the A/D converter, it is the trigger position for the PWM interrupt. When this parameter is 0, the A/D converter (or PWM interrupt) is triggered at the beginning of the PWM cycle. When it is +0.5, the A/D converter (or PWM interrupt) is triggered at the half position of the PWM cycle.
Sync. Mode	Synchronization mode, can be chose from the following: <ul style="list-style-type: none"> - Disabled - Slave (use Sync. i) - Master (use Sync. i) Where i is from 1 to 8
Dead Time	The dead time for the PWM generator, in sec.
Sampling Frequency	Frequency of the PWM generator, in Hz
Peak-to-Peak Value	Peak-to-Peak value of the carrier wave. This parameter is for PWM(sub) only.
Offset Value	DC offset value of the carrier wave. This parameter is for PWM(sub) only.
Initial Rating U, V, W (or A, B, C)	Initial input value of phase u, v, and w for the PWM generator, or phase a, b, and c for PWM(sub) generator
Initial Amplitude	Initial amplitude value of the PWM generator, for Space Vector PWM only.
Initial Angle	Initial angle of the PWM generator, in rad., for Space Vector PWM only.
Start PWM at Beginning	It can be set to either <i>Start</i> or <i>Do not start</i> . When it is set to <i>Start</i> , PWM will start right from the beginning. If it is set to <i>Do not start</i> , one needs to start PWM using the "Start PWM" function.

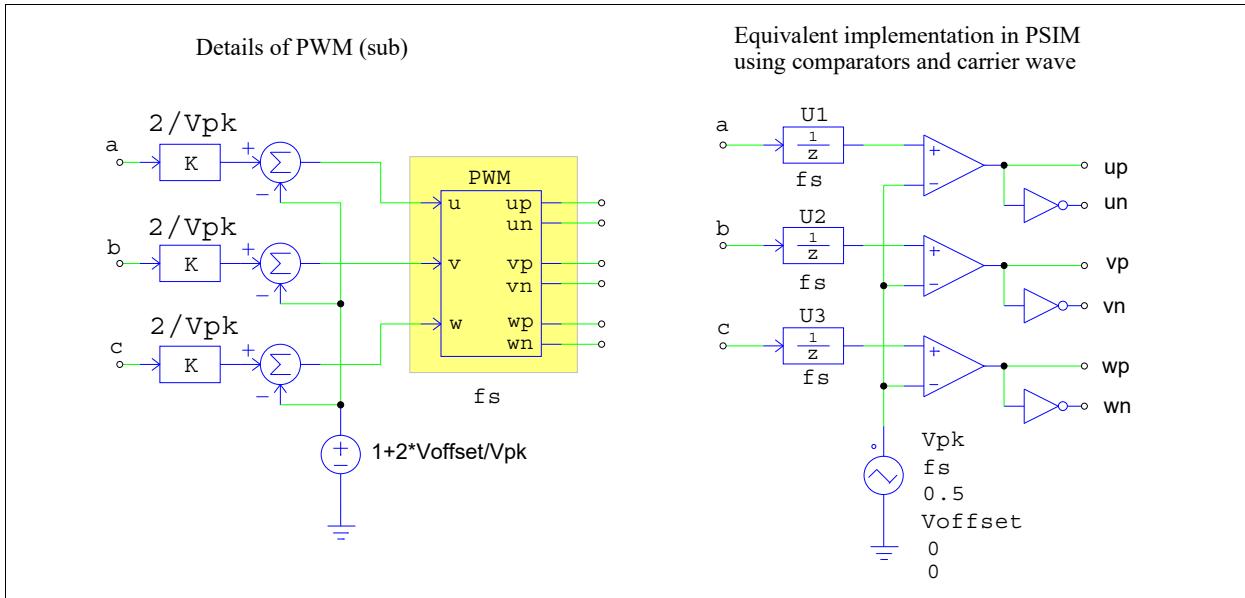
The element **PWM** generates sinusoidal-modulated PWM signals for a three-phase system. The inputs "u", "v", and "w" are for three-phase input modulation signals. The input ranges are between -1 to 1. That is, when the input is -1, the duty cycle is 0, and when the input is 1, the duty cycle is 1. With the input at 0, the duty cycle is 0.5. The carrier wave is a triangular wave with 0.5 duty cycle (the intervals of the rising slope and the falling slope are equal).

The element **Space Vector PWM** generates PWM signals for a three-phase system based on space vector PWM technique. The input "amp" is for the space vector amplitude, and the range is from 0 to 1. The input "ang" is for the space vector phase angle, and the range is from -2π to 4π .

Because the input range of the **PWM** generator is between -1 and 1, while in PSIM simulation, normally PWM signals are generated by comparing a modulation signal with a carrier signal, and the modulation signal range may not be from -1 to 1, scaling may be needed before the modulation signal is sent to the PWM generator.

The element **PWM (sub)** consists of a **PWM** element and the input scaling-offset circuit, as shown below. It

provides convenience to switch from the comparator-based PWM to the hardware PWM generator element



The circuit on the left shows the details of the **PWM (sub)** element. It consists of a scaling-offset circuit and the hardware **PWM** element. With the scaling, ranges of the inputs a , b , and c are no longer limited to -1 and 1. Similar to the definition of a carrier voltage source, the peak-to-peaks value and the dc offset can be defined directly.

The circuit on the right shows the equivalent circuit of the **PWM (sub)** element, implemented in PSIM using comparators and a triangular carrier voltage source. The carrier source parameters are:

V_peak_to_peak:	V_{pk}
Frequency:	f_s
Duty Cycle:	0.5
DC Offset:	V_{offset}
Tstart:	0
Phase Delay:	0

Note the inclusion of three unit delay blocks U1, U2, and U3, as they are used to model the one-cycle delay effect existing in the hardware **PWM** element. Also, the carrier wave duty cycle is fixed at 0.5, as the carrier wave in the hardware **PWM** element is of triangular type.

The Scaling factor specifies how often the PWM-caused-interrupt occurs.

The parameter "Control Frequency" specifies the sampling frequency. This frequency may or may not be the same as the PWM switching frequency.

The relationship between the Control Frequency f_{smp} , the Scaling Factor $Kscale$, and the PWM carrier frequency f_{sw} is calculated as below:

If the parameter "Interrupt Mode" is set as "*Interrupt at peak*" or "*Interrupt at valley*":

$$f_{sw} = Kscale * f_{smp}$$

For example, if $Kscale = 3$, the PWM-caused-interrupt occurs once in every 3 PWM period.

If the parameter "Interrupt Mode" is set as "*Interrupt at peak/valley*":

$$f_{sw} = Kscale * f_{smp}/2$$

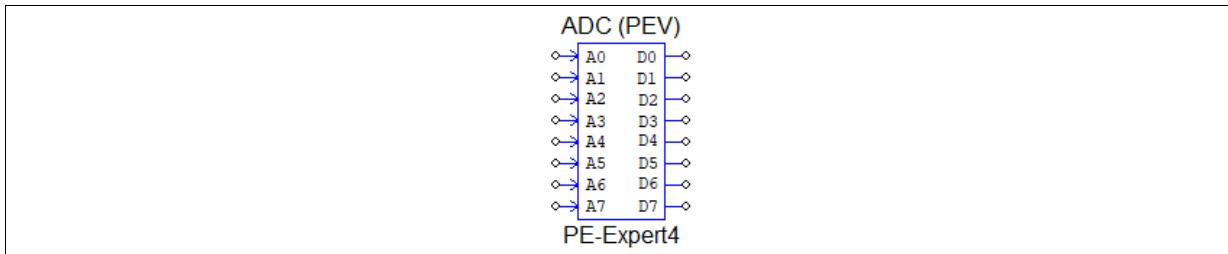
For example, if $Kscale = 3$, the PWM-caused-interrupt occurs once in every $3*2=6$ PWM period.

Please note that when the **PWM** and **PWM (sub)** generators are simulated, the dead time is ignored and is not considered in the simulation.

12.3.3 A/D Converter

An A/D converter converts an analog signal into a digital signal that DSP can process.

Image:



Attributes:

Parameters	Description
Board No.	The board number of the PEV board that contains the A/D converter.
Conversion Mode	ADC conversion mode. It can be either <i>Continuous</i> or <i>PWM triggered</i> .
Ch Ai Output Range	The output range V_{range} of the i_{th} A/D channel ($i = 0$ to 7)
Ch Ai Offset	The output offset V_{offset} of the i_{th} A/D channel ($i = 0$ to 7)

The input range of the A/D converter is from -5V to +5V, and the output range is from $-V_{range}$ to V_{range} . The output is scaled based on the following:

$$V_o = V_i * V_{range} / 5 - V_{offset}$$

For example, if the A/D input $V_i = 2$, $V_{range} = 20$, and $V_{offset} = 0.5$, then

$$V_o = 2 * 20 / 5 - 0.5 = 7.5 \text{ V}$$

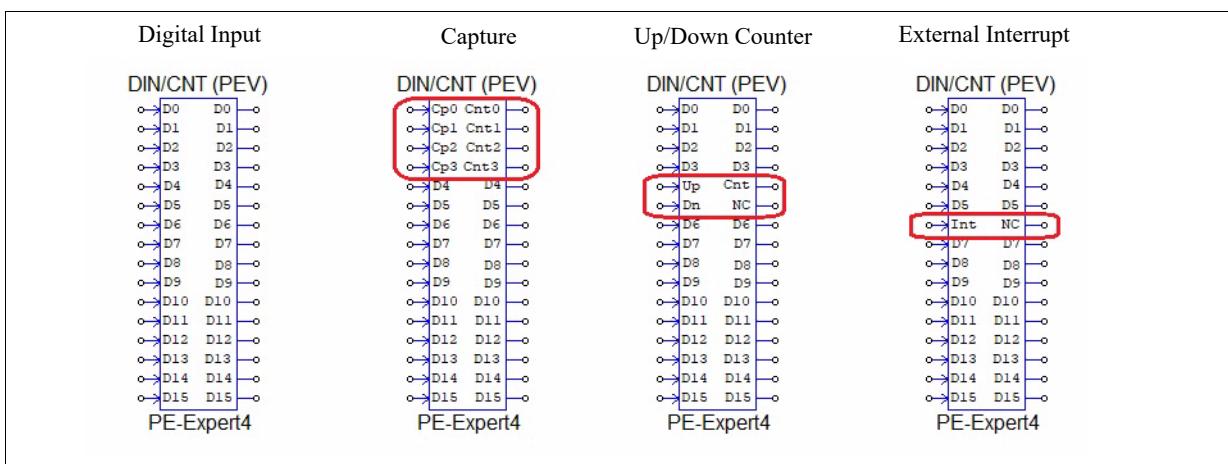
12.3.4 Digital Input / Capture / Counter

The hardware provides a 16-pin digital input element. Note that Pins D0 through D3 are shared with the capture element, Pins D4 and D5 are shared with the up/down counter element, and Pin D6 can be used for input of external interrupt.

Because of the shared pins, a combo element is provided to represent digital input, capture, and counter. Input/output pins are assigned to different functions depending on the definition.

The images and attributes of the combo element in different functions are shown below.

Images:



Attributes:

Parameters	Description
Board No.	The board number of the PEV board that contains the element.
Input/Capture i	The index i changes from 0 to 3, corresponding to Inputs D0 to D3 respectively. The parameter can be defined as one of the following: - <i>Digital Input i</i> : Input pin D_i will be a digital input. - <i>Capture i</i> : Input pin D_i will be the input of Capture i , and the output pin D_i will be the counter output of Capture i . The captions of the input/output pins will be changed to Cap_i and Cnt_i .
Counter Source i	The index i changes from 0 to 3, corresponding to Inputs D0 to D3 respectively. The name of the counter source. The parameter can be either "GP_TIMER" for general-purpose timer, or the name of the encoder.
Input 4 and 5 / Counter	It can be defined as one of the following: - <i>Digital Input 4 and 5</i> : Input pin D4 and D5 will be digital inputs. - <i>Counter</i> : Input pin D4 and D5 will be the inputs of the up/down counter, and the output pin D4 will be the counter output. In the counter mode, the output pin D5 is not used. The captions of the input pin D4 will be changed to <i>Up</i> (for up counter input) and pin D5 to <i>Dn</i> (for down counter input). The caption of the output pin D5 will be changed to <i>Cnt</i> (for counter output), and pin D6 to <i>NC</i> (for not connected).
Counter Mode	When Inputs D4 and D5 are defined as counter inputs, the counter mode can be either <i>Up/down</i> or <i>Direction/pulse</i> .
Input 6 / External Interrupt	It can be one of the following: - <i>Digital Input 6</i> : Input pin 6 will be a digital input. - <i>External Interrupt</i> : This pin will be the input of the external interrupt. The caption of the input will be changed to <i>Int</i> (for interrupt) and the output to <i>NC</i> (for not connected).

As a Capture:

The capture element has 4 inputs. When an input changes from low to high (from 0 to 1), it will capture the counter value of the source, and output it through the output pin. The counter source can be either the general-purpose timer (which is the 32-bit free-run counter on the PEV Board), or the encoder.

As a Counter:

The counter has two modes of operations: up/down mode and direction/pulse mode. When the counter is in the "Up/down" mode, the counter will count up when there is a pulse at the "up" input, and will count down when there is a pulse at the "dn" input.

When the counter is in the "Direction/pulse" mode, and when there is a pulse at the "pulse" input, the counter will count up when the "Dn" input is 0, and will count down when the "Dn" input is 1.

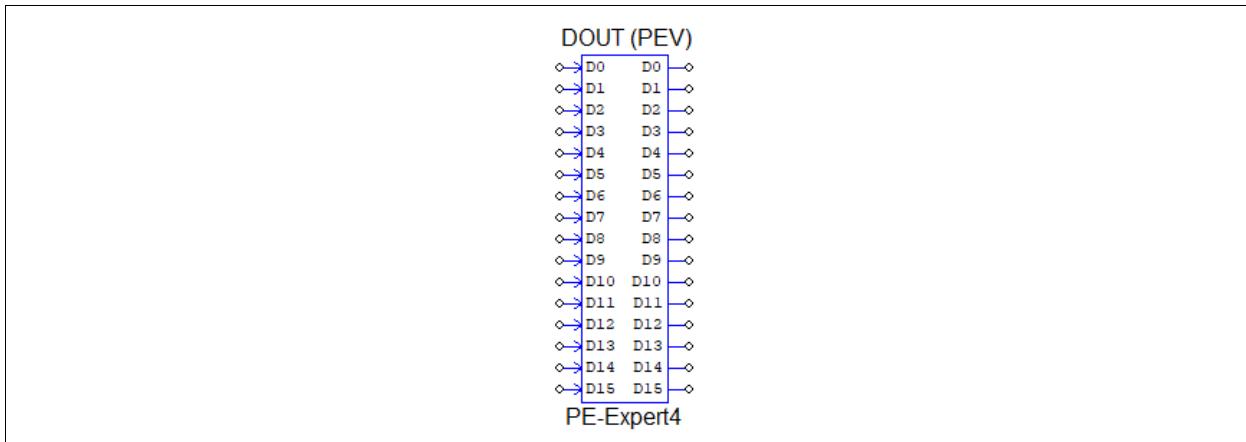
As an External Interrupt:

When Input pin D6 is defined as the input of the external interrupt, when the input changes from 0 to 1, an interrupt will be generated.

12.3.5 Digital Output

The images and attributes of the digital output element are shown below.

Image:



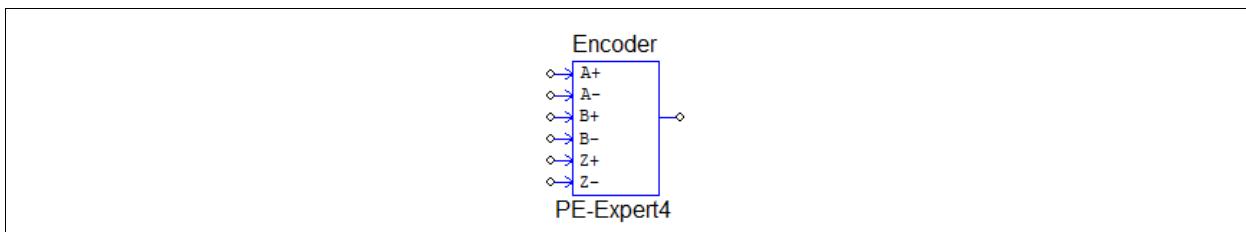
Attributes:

Parameters	Description
Board No.	The board number of the PEV board that contains the element.

12.3.6 Encoder

An encoder is used for position measurement in a motor drive system. It can operate in either "Open Collector" or "Differential Mode" mode.

Image:



Attributes:

Parameters	Description
Board No.	The board number of the PEV board that contains the encoder.
Use Z Signal	Specify if the Z signal is used.
Counting Direction	It can be either <i>Forward</i> or <i>Reverse</i> . When it is set to "Forward", the encoder counts up, and when set to "Reverse", the encoder counts down
Encoder Resolution	Specify encoder resolution. The Encoder counter will count up/down between 1 and Resolution-1.

The output of the encoder output gives the counter value. Also, an interrupt can be generated by the input signal Z+ and Z-.

An encoder can trigger an interrupt by the Z signal. To implement an encoder interrupt routine, one needs to use an interrupt element that specifies the interrupt source (by filling the name of the Encoder element) and which edge triggers interrupt (it can be falling edge, rising edge, or rising/falling edge). This element links to an event subcircuit by an event connection wire.

12.4 PSPWM Board

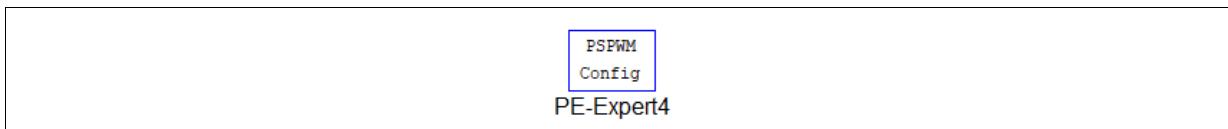
The PSPWM Board contains the following functions and hardware.

- PSPWM-24
- PSPWM-144
- EXGATE-18
- Digital Input
- Digital Output
- PSPWM Board Configuration

12.4.1 PSPWM Board Configuration

This element defines the PSPWM board configuration. If a PSIM circuit users any of the PSPWM board element, this configuration block must be used.

Image:



Attributes:

Parameters	Description
Board No.	The board number of the PEV board that contains these elements (it ranges from 0 to 4).
Interrupt Source of int <i>i</i>	Select the interrupt source that uses int <i>i</i> in the current system.
Interrupt Priority of int <i>i</i>	Select interrupt priority that uses int <i>i</i> in the current system.

This element specifies all interrupt signals used by interrupt sources, for each interrupt signal, one can specify its interrupt priority, too. The priority options are from 4 (highest) to 13 (lowest).

12.4.2 PWM Generators (PSPWM-24, PSPWM-144, and EXGATE-18)

The following elements are for PWM setup.

PSPWM-24 is the 24-channel PWM generator on the PSPWM board for the PE-Expert4 Target

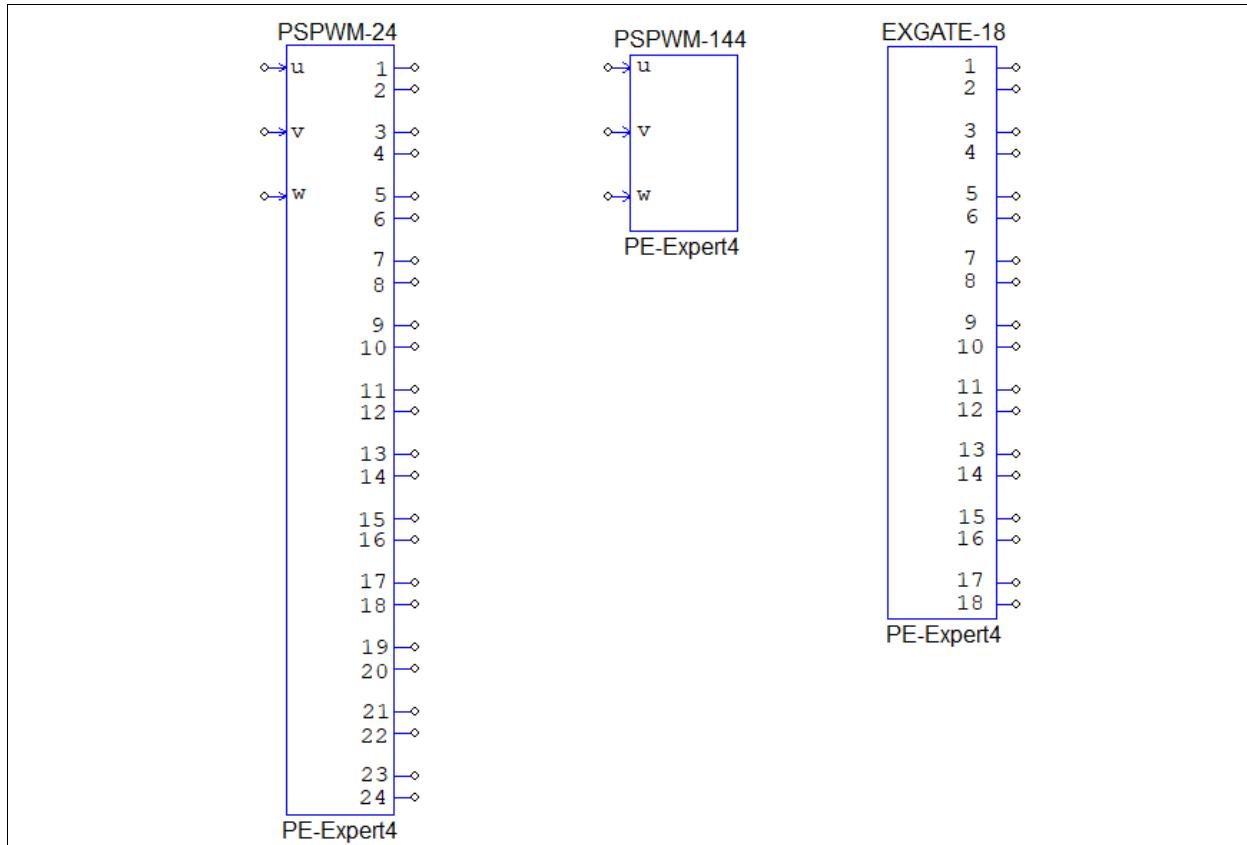
EXGATE-18 is the 18-channel extension boards.

PSPWM-144 is the main FPGA board with connectors for extension boards. It can connect to up to eight of 18-channel extension boards. It is used to set up the PWM for the EXGATE-18 board.

If one or more of EXGATE-18 boards is used in a system, PSPWM-144 must be used to define the PWM parameters.

The images and the parameters for these elements are shown below.

Images:



Attributes of EXGATE-18:

Parameters	Description
Board No.	Board number of the main PSPWM board that supports the extension boards.
Extension Board No	There can be up to 8 PSPWM extension boards. The PSPWM extension board number can be chosen from the drop down list, from 1 to 8.

The settings for the PWM output signals on this board are defined in the element **PSPWM-144** parameters, as listed below.

Attributes of PSPWM-24 and PSPWM-144:

Parameters	Description
Board No.	The board number of the PEV board that contains the PWM generator.
Scaling Factor	Scaling Factor is an integer that ranges from 1 to 1000, specifies the number of PWM period between two PWM-caused-interrupts.
Reference Update Mode	The reference update mode can be set to one of the following: - At each carrier: The real modulation value is applied at each carrier. - At the first carrier: The real modulation value is applied at the first carrier.

Reference Update Time	The reference update time can be set to one of the following: <ul style="list-style-type: none"> - At peak: The new modulation value is applied at each carrier or the first carrier's peak. - At valley: The new modulation value is applied at each carrier or the first carrier's valley. - At peak/valley: The new modulation value is applied at each carrier or the first carrier's peak and valley.
Interrupt Mode	The interrupt mode can be set to one of the following: <ul style="list-style-type: none"> - <i>Interrupt at peak</i>: Interrupt occurs at the peak of the carrier wave - <i>Interrupt at valley</i>: Interrupt occurs at the valley of the carrier wave - <i>Interrupt at peak/valley</i>: Interrupt occurs at both the peak and valley of the carrier wave <p>When the interrupt mode is set to both peak and valley, the interrupt occurs at two times the rate of the carrier wave. In another word, if the carrier frequency is 10kHz, the interrupt rate will be 20kHz.</p>
Interrupt Position	The interrupt position ranges from -0.5 to +0.5. If the value is positive, it is after the beginning of the PWM cycle. If the value is negative, it is before the beginning of the PWM cycle. If the A/D converter is used and its conversion mode is set to 'PWM Triggered', this parameter specifies the position that PWM triggers ADC for conversion. If PWM does not trigger the A/D converter, it is the trigger position for the PWM interrupt. When this parameter is 0, the A/D converter (or PWM interrupt) is triggered at the beginning of the PWM cycle. When it is +0.5, the A/D converter (or PWM interrupt) is triggered at the half position of the PWM cycle.
Number of Converter Levels	Specifies how many level the converter has.
Sync. Mode	Synchronization mode, can be chose from the following: <ul style="list-style-type: none"> - Disabled - Slave(use Sync. <i>i</i>) - Master(use Sync. <i>i</i>) <p>Where <i>i</i> is from 1 to 8</p>
Dead Time	The dead time for the PWM generator, in sec.
Sampling Frequency	Frequency of the PWM generator, in Hz
Operation Mode	The operation mode can be set to one of the following: <ul style="list-style-type: none"> - Diode clamped/Flying cap: Diode clamped/Flying capacitor (two switches per cell) - MMC (chopper): Half bridge cell (two switches per cell) - MMC H-bridge): Full bridge cell (four switches per cell)
Phase Number	The phase number can be set to one of the following: <ul style="list-style-type: none"> - 1 phase with neutral - 1 phase with 2 legs - 3 phase
User H-bridge Shift	Specify if applying 180 degree phase shift between the 2 legs of full bridge cell. This setting is ignored for diode half bridge and clamped/flying capacitor cell.

Shift for Up/Lw Arms	Specify the phase shift between upper arm and lower arm for half bridge or full bridge cell. This setting is ignored for diode clamped/flying capacitor cell.
Start PWM at Beginning	It can be set to either <i>Start</i> or <i>Do not start</i> . When it is set to <i>Start</i> , PWM will start right from the beginning. If it is set to <i>Do not start</i> , one needs to start PWM using the "Start PWM" function.

The PWM generator generates sinusoidal modulated PWM signals for a three-phase system. The inputs "*u*", "*v*", and "*w*" are for three-phase input modulation signals, and the input ranges are decided by parameter "*Peak-to-Peak Value*" and "*Offset Value*".

The maximum input value is

$$(Peak-to-Peak Value)/2 + (Offset Value).$$

The minimum input value is

$$(Offset Value) - (Peak-to-Peak Value) / 2.$$

The PWM carrier waveform is a triangular waveform, with equal rising and falling slope intervals. PSIM assumes that a PWM cycle starts from "valley" of carrier wave. The "peak" of carrier wave is PWM cycle's half way. The frequency of this carrier wave is the same as the PWM switching frequency *fsw*.

Scaling factor specifies how often the PWM-caused-interrupt occurs.

The parameter "Sampling Frequency" specifies the sampling frequency. This frequency may or may not be the same as the PWM switching frequency.

The relationship between the Sampling Frequency *fsmp*, the Scaling Factor *Kscale*, and the PWM carrier frequency *fsw* is calculated as below:

If the parameter "Interrupt Mode" is set as "*Interrupt at peak*" or "*Interrupt at valley*":

$$fsw = Kscale * fsmp$$

For example, if *Kscale* = 3, the PWM-caused-interrupt occurs once in every 3 PWM period.

If the parameter "Interrupt Mode" is set as "*Interrupt at peak/valley*":

$$fsw = Kscale * fsmp/2$$

For example, if *Kscale* = 3, the PWM-caused-interrupt occurs once in every $3*2=6$ PWM period.

The **Synchronization Mode** can be set to master mode or slave mode. Synchronization signal number can be chose from 1 to 8. To synchronize multiple PWMs and/or ADCs in a system, all related PWMs and/or ADCs need to be specified in same synchronization signal number, one PWM is specified as master mode and the others are specified as slave mode.

If this PWM works with an ADC, PSIM generates different target program according to ADC mode. If ADC mode is "Continuous", PWM causes interrupt itself; if ADC mode is "Pwm Triggered", the same board PWM triggers ADC in a certain time specified by parameter "**Interrupt Position**"; if ADC mode is "Trigger by Ext. Sync. #", the PWM set as a synchronization master triggers this ADC in a certain time specified by the parameter "Interrupt Position" in this PWM.

This setting also affects PEV4 board configuration element. If ADC mode is "PWM Triggered", PEV4 board configuration element should set ADC as an interrupt source; otherwise set the PWM that triggers/synchronizes ADC as an interrupt source.

The parameters **Number of Converter Levels**, **Phase Number**, and **Operation Mode** specify the number of PWM outputs, as shown in the table below. There are 2 switches in each Diode clamped/flying cap and MMC (Chopper) cell, 4 switches in each H-bridge cell.

24 Channel PWM Generator (PSPWM Board) Output Signals

Conversion Level	Phase Number	PWM outputs for 2sw	PWM outputs for 4sw
3	1 phase with neutral	4	8
	1 phase with 2 legs	8	16
	3 phases	12	24
5	1 phase with neutral	8	16
	1 phase with 2 legs	16	N/A
	3 phases	24	N/A
7	1 phase with neutral	12	24
	1 phase with 2 legs	24	N/A
	3 phases	N/A	N/A
9	1 phase with neutral	16	N/A
	1 phase with 2 legs	N/A	N/A
	3 phases	N/A	N/A
11	1 phase with neutral	20	N/A
	1 phase with 2 legs	N/A	N/A
	3 phases	N/A	N/A
13	1 phase with neutral	24	N/A
	1 phase with 2 legs	N/A	N/A
	3 phases	N/A	N/A

PWM Generator (PSPWM-144 Board) Output Signals

Conversion Level	Phase Number	PWM outputs for 2sw	PWM outputs for 4sw
3	1 phase with neutral	4	8
	1 phase with 2 legs	8	16
	3 phases	12	24
5	1 phase with neutral	8	16
	1 phase with 2 legs	16	32
	3 phases	24	48
7	1 phase with neutral	12	24
	1 phase with 2 legs	24	48
	3 phases	36	72
9	1 phase with neutral	16	32
	1 phase with 2 legs	32	64
	3 phases	48	96
11	1 phase with neutral	20	40
	1 phase with 2 legs	40	80
	3 phases	60	120
13	1 phase with neutral	24	48
	1 phase with 2 legs	48	96
	3 phases	72	144

PWM outputs are assigned from the first output pin, phase U outputs are assigned, next phase V then phase W. For example, for 3-phase, 5-level and half bridge cell, PWM outputs are assigned as follows:

U1P, U1N, U2P, U2N, U3P, U3N, U4P, U4N,
 V1P, V1N, V2P, V2N, V3P, V3N, V4P, V4N,
 W1P, W1N, W2P, W2N, W3P, W3N, W4P, W4N

where

U1P connects to S1p of phase U's first cell,
U1N connects to S1n of phase U's first cell.

For 1-phase, 5-level and full bridge cell, PWM outputs are assigned as follows:

U1P1, U1N1, U1P2, U1N2,
U2P1, U2N1, U2P2, U2N2,
U3P1, U3N1, U3P2, U3N2,
U4P1, U4N1, U4P2, U4N2,

the rest 8 outputs are not used.

Here

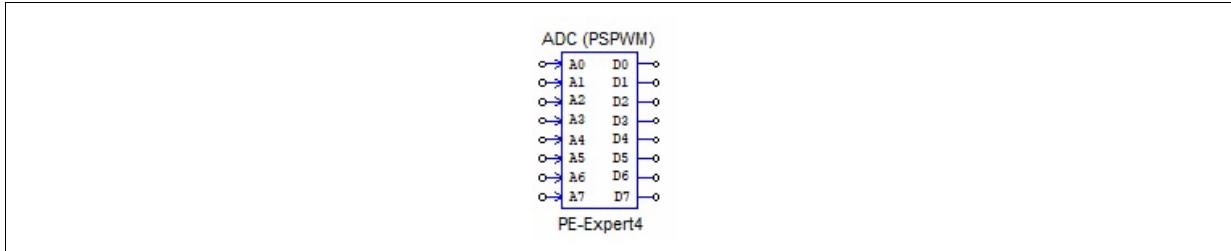
U1P1 connects to S1p of phase U's first cell,
U1N2 connects to S1n of phase U's first cell,
U1P2 connects to S2p of phase U's first cell,
U1N2 connects to S2n of phase U's first cell.

To start PWM, apply a high logic signal (1) to the input of the "Start PWM" element. To stop PWM, apply a high logic signal (1) to the input of the "Stop PWM" element.

12.4.3 A/D Converter

An A/D converter converts an analog signal into a digital signal that DSP can process.

Image:



Attributes:

Parameters	Description
Board No.	The board number of the PEV board that contains the A/D converter.
Conversion Mode	ADC conversion mode. It can be either <i>Continuous</i> or <i>PWM triggered</i> .
Ch Ai Range	The output range V_{range} of the i_{th} A/D channel ($i = 0$ to 7)
Ch Ai Offset	The output offset V_{offset} of the i_{th} A/D channel ($i = 0$ to 7)

The input range of the A/D converter is from -5V to +5V, and the output range is from $-V_{range}$ to V_{range} . The output is scaled based on the following:

$$V_o = V_i * V_{range} / 5 - V_{offset}$$

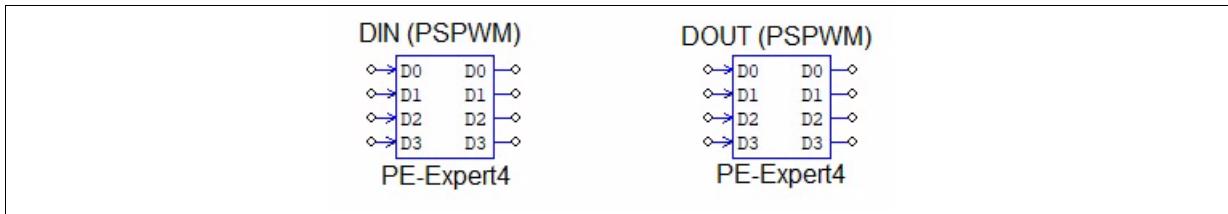
For example, if the A/D input $V_i = 2$, $V_{range} = 20$, and $V_{offset} = 0.5$, then $V_o = 2 * 20 / 5 - 0.5 = 7.5$ V

This A/D converter can not be triggered by synchronization signal. It can be only triggered by the same PSPWM board.

12.4.4 Digital Input and Output

The images and attributes of the digital input and output elements are shown below.

Image:



Attributes:

Parameters	Description
Board No.	The board number of the PSPWM board that contains the element.

12.5 ADC Board

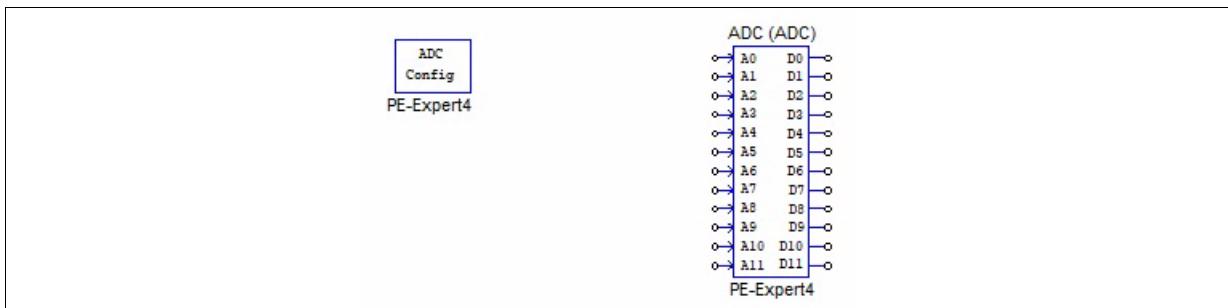
The MSPE4-ADC board has an A/D converter of 12 channels with 14-bit resolution. This ADC converts analog signals into digital signal that DSP can process.

Two elements available to set up this board:

- ADC Board Configuration
- ADC Converter

The images and the parameters of these elements are shown below.

Images:



Attributes of ADC Board Configuration:

Parameters	Description
Board No.	The board number of the PEV board that contains these elements.
Interrupt Source of inti	Specify interrupt source that uses inti in the current system.
Interrupt Priority of inti	Specify interrupt priority that uses inti in the current system.

This element specifies all interrupt signals used by interrupt sources, for each interrupt signal, one can specify its interrupt priority, too.

Attributes of ADC Converter:

Parameters	Description
Board No.	The board number of the PEV board that contains the A/D converter.

Conversion Mode	ADC conversion mode. It can be either <i>Continuous</i> or <i>Trigger by Ext. Sync. i</i> , where i is from 1 to 8.
Ch A_i Range	The output range V_{range} of the i_{th} A/D channel ($i = 0$ to 11)
Ch A_i Offset	The output offset V_{offset} of the i_{th} A/D channel ($i = 0$ to 11)

The input range of the A/D converter is from -5V to +5V, and the output range is from $-V_{range}$ to V_{range} . The output is scaled based on the following:

$$V_o = V_i * V_{range} / 5 - V_{offset}$$

For example, if the A/D input $V_i = 2$, $V_{range} = 20$, and $V_{offset} = 0.5$, then $V_o = 2 * 20 / 5 - 0.5 = 7.5$ V

12.6 PE-Expert4 Runtime Library Functions

PE-Expert4 provides a runtime library for high-speed calculation. When the code is generated, whenever possible, functions from the PE-Expert4 runtime library are used.

The table below shows the PSIM elements and the corresponding PE-Expert4 runtime library functions.

PSIM Elements	Runtime Library Functions
Sine or Sine (in rad.)	mwsin (float x)
Cosine or Cosine (in rad.)	mwcoss (float x)
Tangent Inverse	mwarctan2 (float y, float x)
Square-root	mwsqrt (float x)
abc-alpha/beta Transformation	uvw2ab (float u, float v, float w, float *a, float *b)
ab-alpha/beta Transformation	uv2ab (float u, float v, float *a, float *b)
ac-alpha/beta Transformation	uw2ab (float u, float w, float *a, float *b)
alpha/beta-abc Transformation	ab2uvw (float a, float b, float *u, float *v, float *w)
alpha/beta-dq Transformation	ab2dq (float a, float b, float *d, float *q)
dq-alpha/beta Transformation	dq2ab (float d, float q, float *a, float *b)
xy-r/angle Transformation	xy2ra (float y, float x, float *a, float *b)
r/angle-xy Transformation	ra2zy (float r, float a, float *x, float *y)

13.1 Overview

The SimCoder library supports the TI Digital Motor Control (DMC) library versions V4.0, V4.1 and V4.2. User may select the version in the **Simulation Control** under the **SimCoder** tab. Note that only one version can be used at a time. Once a version is selected, elements that are not supported in that version will be disabled.

SimCoder's TI DMC Library contains the function blocks listed in the table below. A brief description of these blocks are given in this Chapter. For more detailed description, please refer to the document from Texas Instruments.

Element Name	Description	Version Available		
ACI_FE	Flux estimator of the 3-phase induction motor	4.0	4.1	4.2
ACI_SE	Speed estimator of the 3-phase induction motor	4.0	4.1	4.2
ANGLE_MATH	Wrap angle within 0 and 1 (0 to 2π) or -0.5 and 0.5 (- π to $+\pi$)			4.2
CLARK	Clarke transformation	4.0	4.1	4.2
COMTN_TRIG	Commutation trigger generator	4.0	4.1	4.2
CUR_MOD	Current model	4.0	4.1	4.2
IMPULSE	Impulse generator	4.0	4.1	4.2
IPARK	Inverse Park transformation	4.0	4.1	4.2
PARK	Park transformation	4.0	4.1	4.2
PHASE_VOLT	Phase voltage reconstruction	4.0	4.1	4.2
PI	PI controller with anti-windup correction	4.0	4.1	4.2
PI_POS	PI controller with anti-windup and position error wrapper		4.1	4.2
PI_POS_REG4	PI controller with anti-windup and position error wrapper, proportional gain separated.			4.2
PI_REG4	PI controller with anti-windup correction, proportional gain separated.			4.2
PID_GRANDO	PID controller grando	4.0	4.1	4.2
PID_REG3	Digital PID controller with anti-windup	4.0	4.1	4.2
RAMP_GEN	Generate a ramp with adjustable gain	4.0	4.1	4.2
RMP_CNTL	Ramp up/down to targeted value and flag when output equals input.	4.0	4.1	4.2
RMP2_CNTL	Ramp up/down to targeted value	4.0	4.1	4.2
RMP3_CNTL	Ramp down to targeted value and flag when output equals input	4.0	4.1	4.2
SMOPOS	Sliding-mode rotor position observer of PMSM	4.0	4.1	4.2

SPEED_EST	Speed calculator based on rotor angle without direction information	4.0	4.1	4.2
SPEED_FR	Speed calculator based on rotor angle from encoder sensor	4.0	4.1	4.2
SPEED_PRD	Speed calculator based on period measurement	4.0	4.1	4.2
SVGEN	Space vector generator with quadrature control	4.0	4.1	4.2
SVGEN_COMM	Space vector generator using common mode voltage	4.0	4.1	4.2
SVGEN_DPWM	Space vector generator with DPWM approach	4.0	4.1	4.2
SVGEN_MF	Space vector generator using magnitude and frequency	4.0	4.1	4.2
VHZ_PROFILE	Volt/Hertz profile for ac induction motor	4.0	4.1	4.2

13.2 ACI_FE: Flux Estimator of 3-phase Induction Motors

This block implements the flux estimator with the rotor flux angle for the 3-phase induction motor based upon the integral of back emf's (voltage model) approach. To reduce the errors due to pure integrator and stator resistance measurement, the compensated voltages produced by PI compensators are introduced. Therefore, this flux estimator can be operating over a wide range of speed, even at very low speed.

The ACI_FE module requires eight constants according to the machine parameters, base quantities, mechanical parameters, and sampling period. These eight constants are computed by the macro ACI_FE_CONST.

SimCoder combined ACI_FE and ACI_FE_CONST functions into one block to simplifies the process. When this element is in a circuit schematic, SimCoder copies ACI_FE macro to the generated project folder and the ACI_FE_CONST sub-module in ACI_FE's initial data structure definition.

The overall of the flux estimator is shown below. The rotor flux linkages in the stationary reference frame are mainly computed by means of the integral of back emf's in the voltage model. By introducing the compensated voltages generated by PI compensators, the errors associated with pure integrator and stator resistance measurement can be taken care.

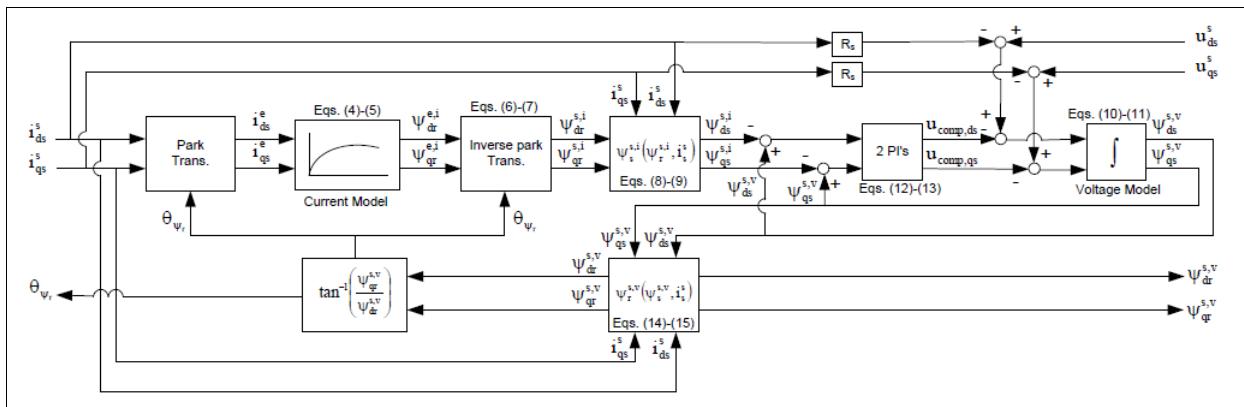
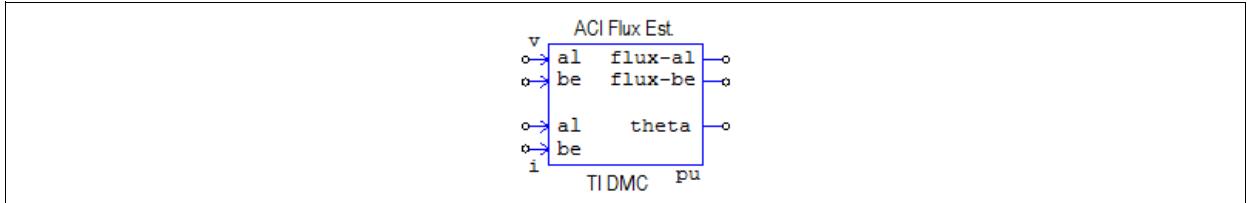


Image:



Attributes:

Parameters	Description
Stator Resistance Rs	Motor stator resistance, in ohm
Stator Inductance Ls	Motor stator inductance, in H
Rotor Resistance Rr	Motor rotor resistance, in ohm
Rotor Inductance Lr	Motor rotor inductance, in H
Magnetizing Inductance Lm	Motor magnetizing inductance, in H
Base Phase Voltage Vb	Motor base phase voltage, in volt
Base Phase Current Ib	Motor base phase current, in ampere
PI Proportional Gain Kp	PI controller proportional gain
PI Integral Gain Ki	PI controller integral gain
Sampling Frequency	System sampling frequency

Input and Output Signals:

Name	I/O	Description
ud	Input	Stationary d-axis stator voltage
uq	Input	Stationary q-axis stator voltage
id	Input	Stationary d-axis stator current
iq	Input	Stationary q-axis stator current
flux-d	Output	Stationary d-axis estimated rotor flux
flux-q	Output	Stationary q-axis estimated rotor flux
theta	Output	Rotor flux angle, in degrees, range 0 to 360

13.3 ACI_SE: Speed Estimator of 3-phase Induction Motors

This block implements a speed estimator of the 3-phase motor based upon its mathematics model. The estimator's accuracy relies heavily on knowledge of critical motor parameters.

The speed estimator of Induction motor module requires four constants according to the machine parameters, base quantities, mechanical parameters, and sampling period. These four constants are computed by the macro ACI_SE_CONST.

SimCoder combined ACISE and ACISE_CONST in this block to simplify the process. When this element is in a circuit schematic, SimCoder copies ACISE macro to the generated project folder and implemented ACISE_CONST sub-module in ACI_SE's initial data structure definition.

The open-loop speed estimator is derived basing on the mathematics equations of induction motor in the stationary reference frame. The precise values of machine parameters are unavoidably required, otherwise the steady-state speed error may happen.

Three equations are mainly employed to compute the estimated speed in per unit:

The rotor speed in per unit:

$$\omega_{r,pu} = \omega_{e,pu} - K_1 \left(\frac{\lambda_{dr,pu}^s i_{qs,pu}^s - \lambda_{qr,pu}^s i_{ds,pu}^s}{(\lambda_{r,pu}^s)^2} \right) \text{ pu}$$

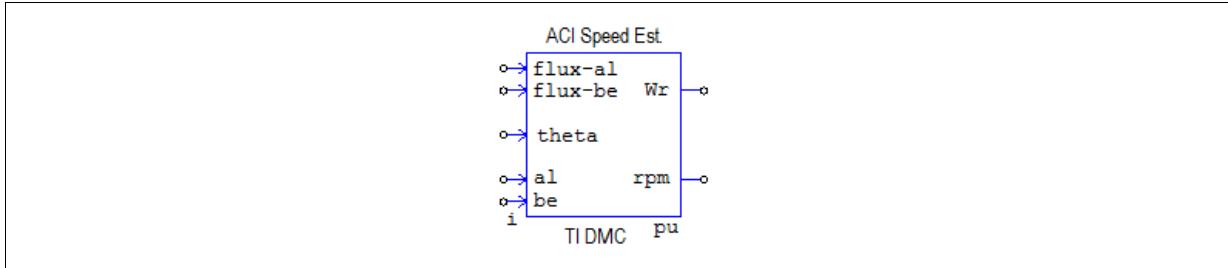
The synchronous speed in per unit:

$$\omega_{e,pu}(k) = K_2 (\theta_{\lambda_r,pu}(k) - \theta_{\lambda_r,pu}(k-1)) \text{ pu}$$

The estimated speed in per unit:

$$\hat{\omega}_{e,pu}(k) = K_3 \hat{\omega}_{e,pu}(k-1) + K_4 \omega_{e,pu}(k) \text{ pu}$$

Image:



Attributes:

Parameters	Description
Rotor Resistance Rr	Motor rotor resistance, in ohm
Rotor Inductance Lr	Motor rotor inductance, in H
Filter Cut-off Frequency fc	Cut-off frequency of the low pass filter, in Hz
Base Electrical Frequency fb	Base electrical frequency, in Hz
Base Speed in rpm	Base motor speed in rpm
Sampling Frequency	System sampling frequency

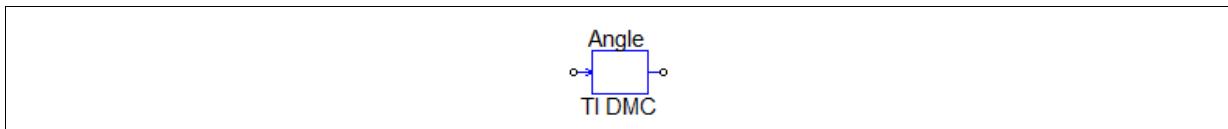
Input and Output Signals:

Name	I/O	Description
flux-d	Input	Stationary d-axis estimated rotor flux
flux-q	Input	Stationary q-axis estimated rotor flux
theta	Input	Rotor flux angle, in degrees, range 0 to 360
id	Input	Stationary d-axis stator current
iq	Input	Stationary q-axis stator current
Wr	Output	Estimated motor speed in per unit
rpm	Output	Estimated motor speed in rpm

13.4 ANGLE_MATH: Angle Wrap

This block wraps angle within 0 and 1.0, or wraps angle within -0.5 and +0.5.

Image:



Attributes:

Parameters	Description
Angle Range	Define the range of angle output. 0 to +1.0: wrap angle within 0 and 2π (1.0) -0.5 to +0.5: wrap angle within $-\pi$ (-0.5) and $+\pi$ (+0.5)

Input and Output Signals:

Name	I/O	Description
	Input	Raw angle input
	Output	Wrapped angle output

13.5 CLARKE: Clarke Transformation

This block implements the Clarke transformation from balanced 3-phase quantities into balanced 2-phase quadrature quantities.

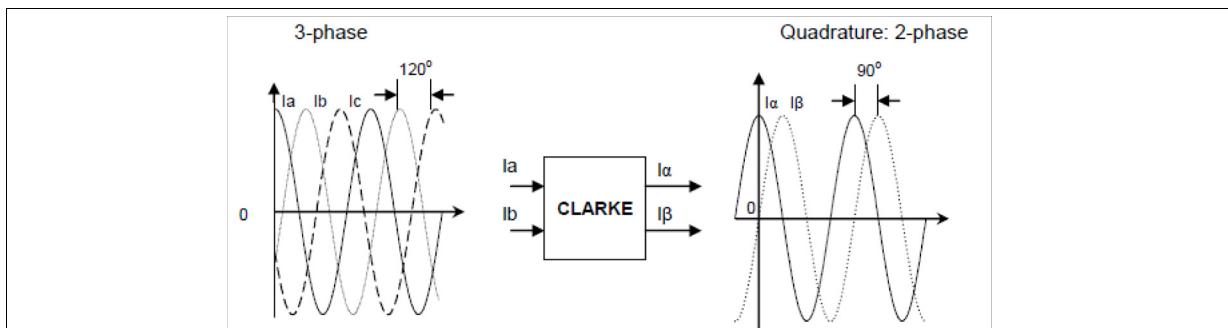
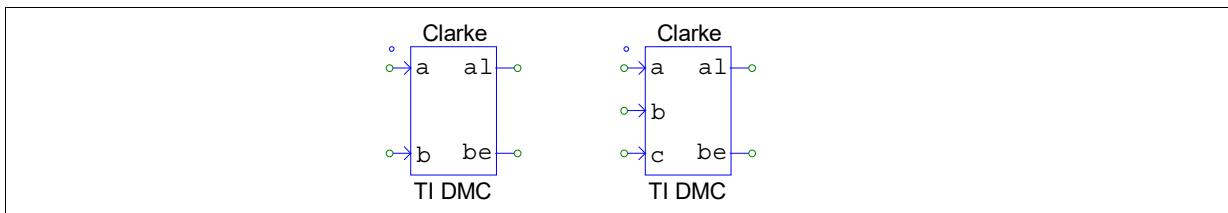


Image:



Attributes:

Parameters	Description
Number of Input Phase	Define the number of input phases. - 2-Phase: Inputs are Phases a and b; - 3-phase: Inputs are Phases a, b, and c.

Input and Output Signals:

Name	I/O	Description
a	Input	Phase A component of the balanced 3-phase quantities.
b	Input	Phase B component of the balanced 3-phase quantities.
al	Output	Direct axis (d) component of the transformed signal.
be	Output	Quadrature axis (q) component of the transformed signal.

The transformation equations are given below:

If the number of input phase is selected as 2-Phase, the transformation equations are:

$$v_\alpha = v_a$$

$$v_\beta = \frac{2v_b + v_a}{\sqrt{3}}$$

If the number of input phase is selected as 3-Phase, the transformation equations are:

$$v_\alpha = v_a$$

$$v_\beta = \frac{v_b - v_c}{\sqrt{3}}$$

13.6 COMTN_TRIGGER: Commutation Trigger Generator for BLDC Motors

This block determines the back emf zero crossing points of a 3-phase BLDC motor based on motor phase voltage measurements and then generates the commutation trigger points for the 3-phase power inverter switches.

The figure below shows the 3-phase power inverter topology used to drive a 3-phase BLDC motor. In this arrangement, the motor and inverter operation is characterized by a two phase ON operation. This means that two of the three phases are always energized, while the third phase is turned off.

The bold arrows on the wires indicate the Direct Current flowing through two motor stator phases. For sensorless control of BLDC drives it is necessary to determine the zero crossing points of the three Bemf voltages and then generate the commutation trigger points for the associated 3-ph power inverter switches.

The COMTN_TRIGGER module computes the neutral voltage, The back emf zero-crossing point, and the time delay corresponding to the commutation delay angle.

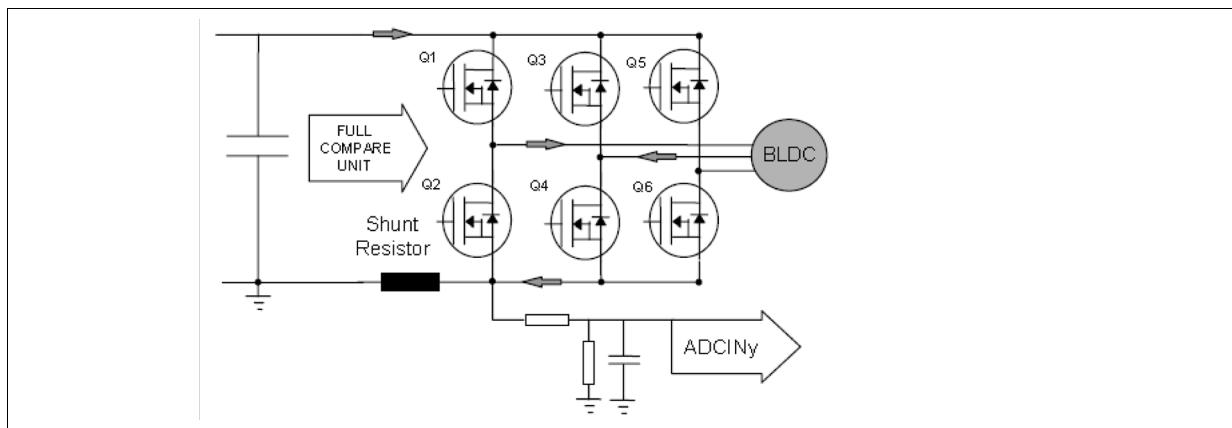
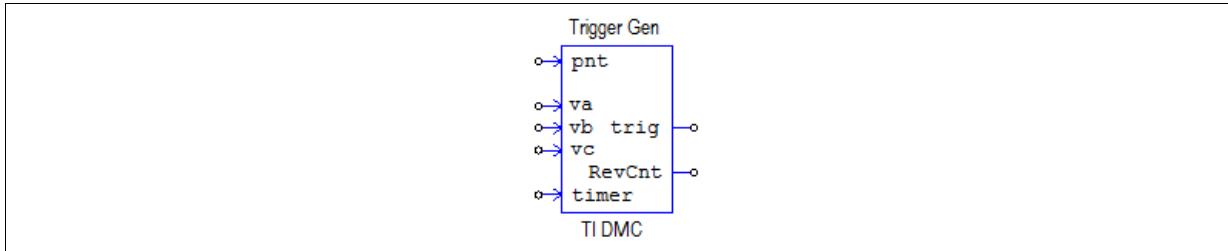


Image:**Attributes:**

Parameters	Description
Noise Windows Delta	Noise windows delta
Noise Windows Threshold	Noise windows dynamic threshold

Input and Output Signals:

Name	I/O	Description
pnt	Input	Commutation state pointer
va	Input	Motor phase a voltage referenced to ground
vb	Input	Motor phase b voltage referenced to ground
vc	Input	Motor phase c voltage referenced to ground
time	Input	Virtual timer for commutation delay angle calculation
trig	Output	Commutation trigger output, (0 or 0x7FFF)

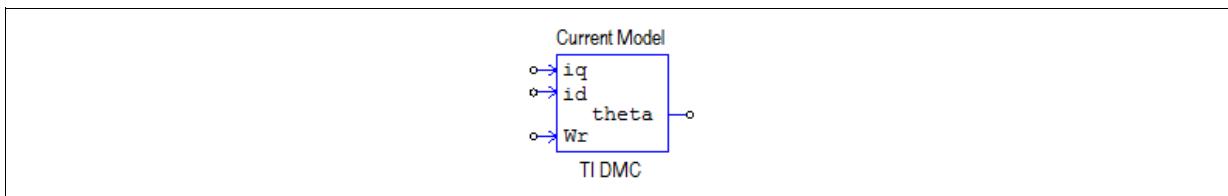
13.7 CUR_MOD: Current Model

This block calculates the rotor flux position from motor current measurements after Park transformation.

With the asynchronous drive, the mechanical rotor angular speed is not equal to the rotor flux angular speed. This implies that the necessary rotor flux position cannot be detected directly by the mechanical position sensor used with the asynchronous motor (QEP or tachometer). The current model module is usually added to the generic structure in the regulation block diagram to perform a current and speed closed loop for a three phases ACI motor in FOC control.

This module requires three constants according to machine parameters, base quantities, mechanical parameters, and sampling period. These constants are computed by the macro CUR_MOD_CONST.

SimCoder combines the functions of CUR_MOD and CUR-MOD_CONST to simplify the process. When this element is used in a circuit schematic, SimCoder copies CUR_MOD macro into the generated project code folder and implements CUR_MOD_CONST sub module in CUR_MOD's initial data structure definition.

Image:

Attributes:

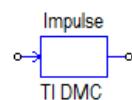
Parameters	Description
Rotor Resistance Rr	Motor rotor resistance, in ohm
Rotor Inductance Lr	Motor rotor inductance, in H
Base Electrical Frequency	Base electrical frequency in Hz
Sampling Frequency	System sampling frequency

Input and Output Signals:

Name	I/O	Description
iq	Input	Synchronous rotating q-axis current
id	Input	Synchronous rotating d-axis current
Wr	Input	Motor rotor electrical angular velocity
theta	Output	Motor rotor flux angle, in degree, range 0 to 360

13.8 IMPULSE: Impulse Generator

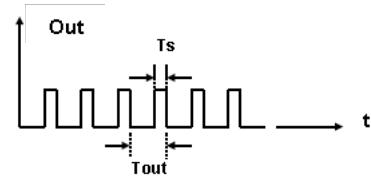
This block generates a periodic impulse.

Image:**Input and Output Signals:**

Name	I/O	Description
	Input	Period of output in number of sampling period
	Output	Impulse output.

If sampling time period is Ts and input = N, then, the output period Tout = N times sampling period.

Out = 0x7FFF for one sampling period at the end of Tout.
Out = 0 for the rest of the time.



13.9 IPARK: Inverse Park Transformation

This block implements the transformation vectors in orthogonal rotating reference frame into two phase orthogonal stationary frame, as shown below:

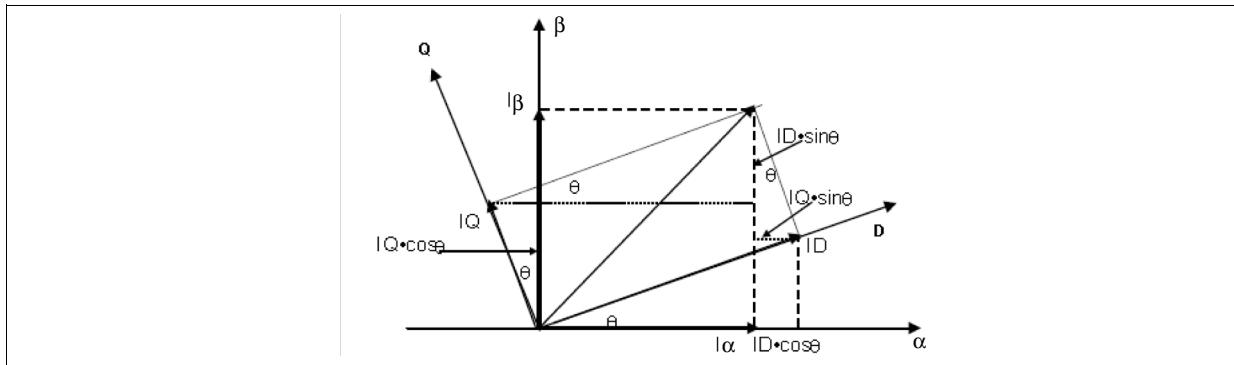
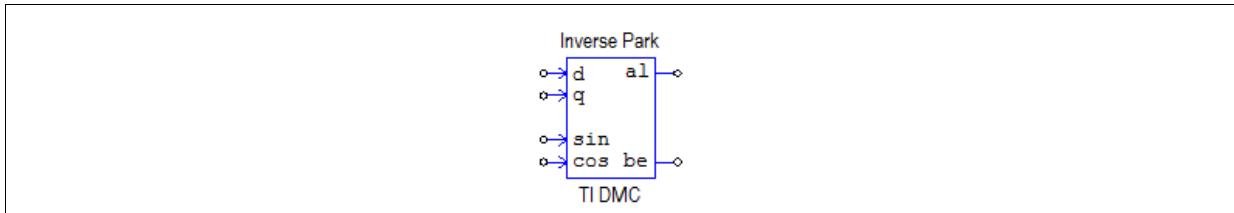


Image:



Input and Output Signals:

Name	I/O	Description
d	Input	Rotating d-axis stator variable
q	Input	Rotating q-axis stator variable
sin	Input	SIN of the phase angle between stationary and rotating frame
cos	Input	COS of the phase angle between stationary and rotating frame
al	Output	Stationary d-axis stator variable
be	Output	Stationary q-axis stator variable

The transformation equations are:

$$v_\alpha = v_d \cdot \cos\theta - v_q \cdot \sin\theta$$

$$v_\beta = v_d \cdot \sin\theta + v_q \cdot \cos\theta$$

13.10 PARK: Park Transformation

This block implements the transformation which converts vectors in balanced 2-phase orthogonal stationary system into orthogonal rotating reverence frame, as shown below.

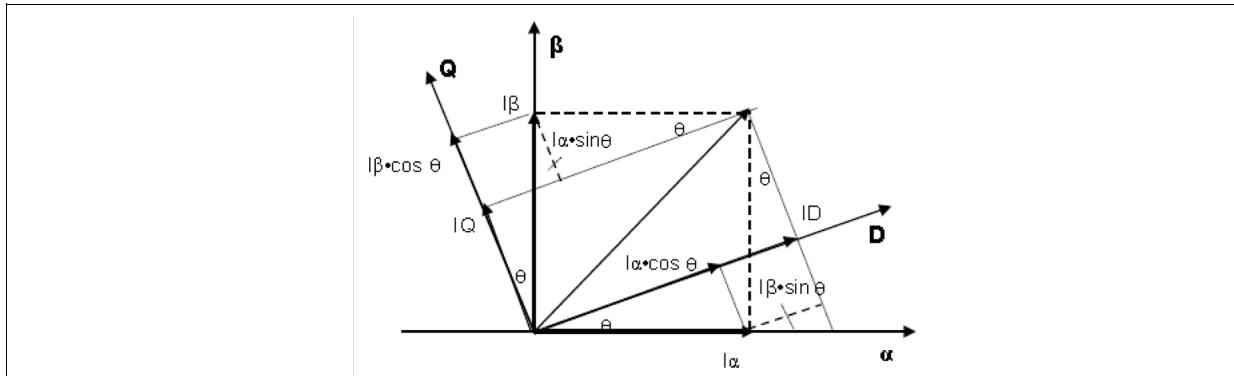
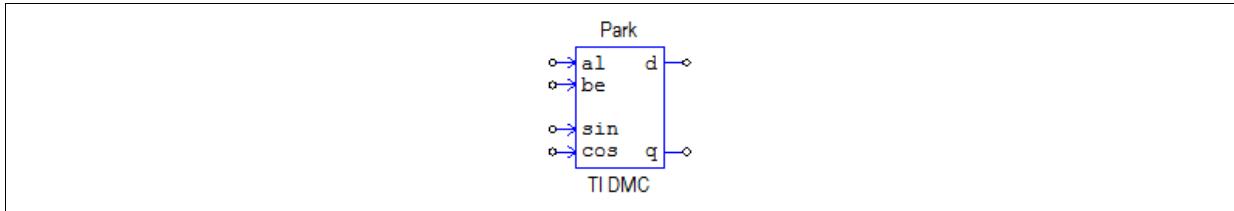


Image:



Input and Output Signals:

Name	I/O	Description
al	Input	Stationary d-axis stator variable v_α
be	Input	Stationary q-axis stator variable v_β
sin	Input	SIN of the phase angle between stationary and rotating frame
cos	Input	COS of the phase angle between stationary and rotating frame
d	Output	Rotating d-axis stator variable v_d
q	Output	Rotating q-axis stator variable v_q

The transformation equations are:

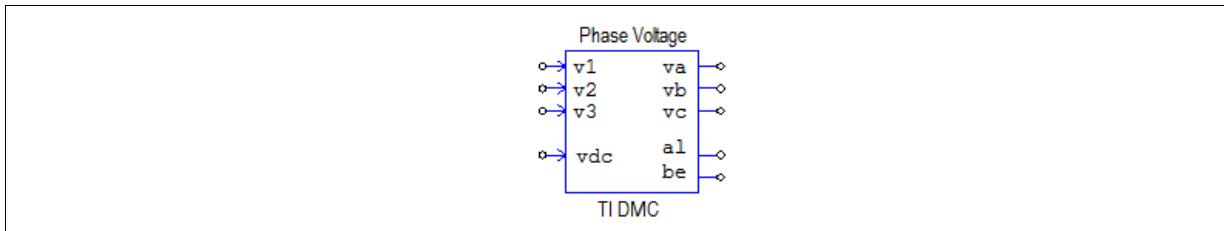
$$v_d = v_\alpha \cdot \cos\theta + v_\beta \cdot \sin\theta$$

$$v_q = v_\alpha \cdot \sin\theta - v_\beta \cdot \cos\theta$$

13.11 PHASE_VOLT: Phase Voltage Reconstruction

This element calculates three phase voltages impressing to the 3-phase electric motor (i.e., induction or synchronous motor) by using the conventional voltage-source inverter. Three phase voltages can be reconstructed from the DC-bus voltage and three switching functions of the upper power switching devices in the inverter. In addition, this software module also includes the clarke transformation changing from three phase voltages into two stationary dq-axis phase voltages.

Image:



Attributes:

Parameters	Description
Out-of-Phase Adjustment	Specify if the input phase signals are out of phase with respect of output. - Yes if 3-phase input signals are from the lower switching functions. - No if 3-phase input signals are from upper switching functions

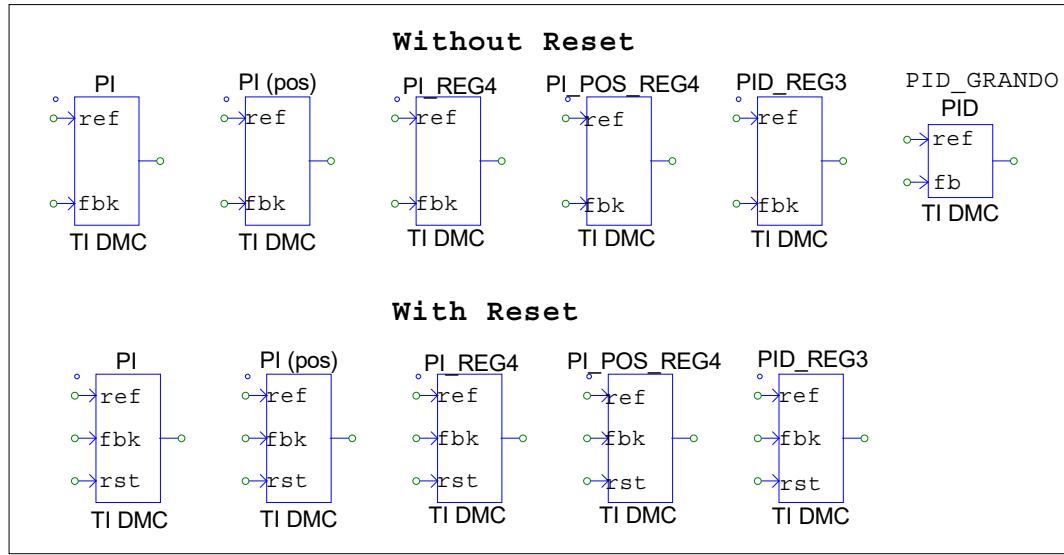
Input and Output Signals:

Name	I/O	Description
v1, v2, v3	Inputs	Modulation voltage phase A, B, and C for inverter upper switching devices.
vdc	Input	DC bus voltage
va, vb, vc	Outputs	Line-to-neutral voltage for phase A, B, and C
al	Output	Stationary d-axis phase voltage
be	Output	Stationary q-axis phase voltage

13.12 PID Controllers

SimCoder implemented all the TI DMC PID controller algorithm into different function blocks. The images and the input and output signals of these PID controllers are show below. All controllers except PID_GRANDO allow user to select with or without reset. If *With Reset* is selected, an extra input *rst* will be added to the block. The integrator output will be clamped to zero when the *rst* input is logically high.

Images:



Input and Output Signals:

Name	I/O	Description
ref	Input	Reference set point
fb	Input	Feedback
rst	Input	When set to logic high (1), the integral output is clamped to zero.
	Output	Controller output

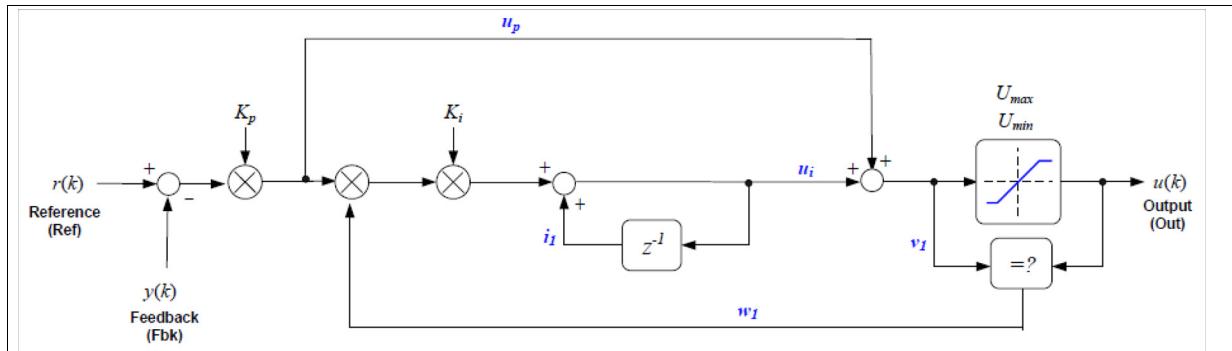
The functions of these controllers are described in the subsections.

13.12.1 PI: PI Controller with Anti-Windup

This block implements a simple 32-bit digital PI controller with anti-windup correction. It consists of a basic summing junction and P+I control law with the following features:

- Programmable output saturation
- Independent reference weighting on proportional path
- Anti-windup integrator reset.

The PI controller is a sub-set of the PID controller. All input, output and internal data are in IQ24 fixed-point format. The block diagram of the internal controller structure is shown below.



Attributes:

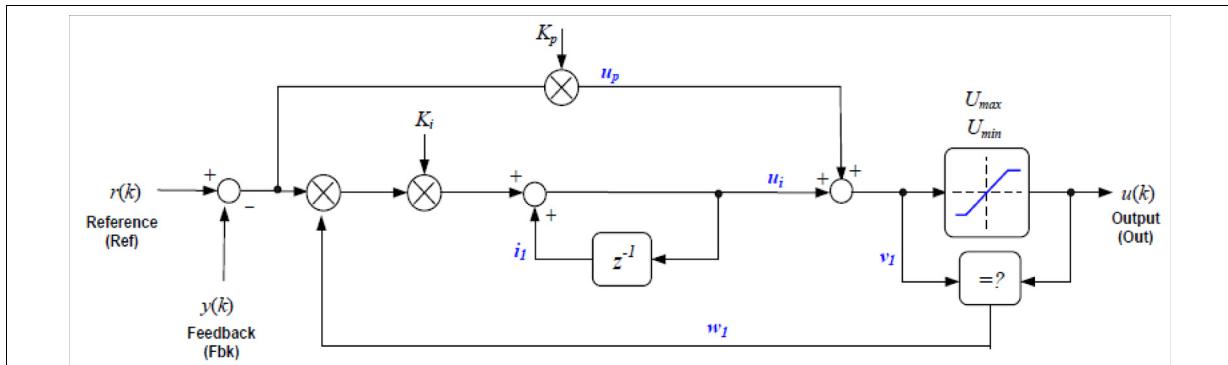
Parameters	Description
Proportional Gain Kp	Proportional loop gain
Integral Gain Ki	Integral gain
Maximum Output	Maximum output limit Umax
Minimum Output	Minimum output limit Umin

13.12.2 PI_REG4: PI Controller with Anti-Windup

This module implements a simple 32-bit digital PI controller with anti-windup correction. Functionally, it is similar to PI module described above, the difference is in the path of P control such that Kp can be set to zero. It consists of a basic summing junction and P+I control law with the following features:

- Programmable output saturation
- Independent reference weighting on proportional path
- Anti-windup integrator reset.

The PI_POS controller is a sub-set of the PID controller. All input, output and internal data are in IQ24 fixed-point format. The block diagram of the internal controller structure is shown below.



Attributes:

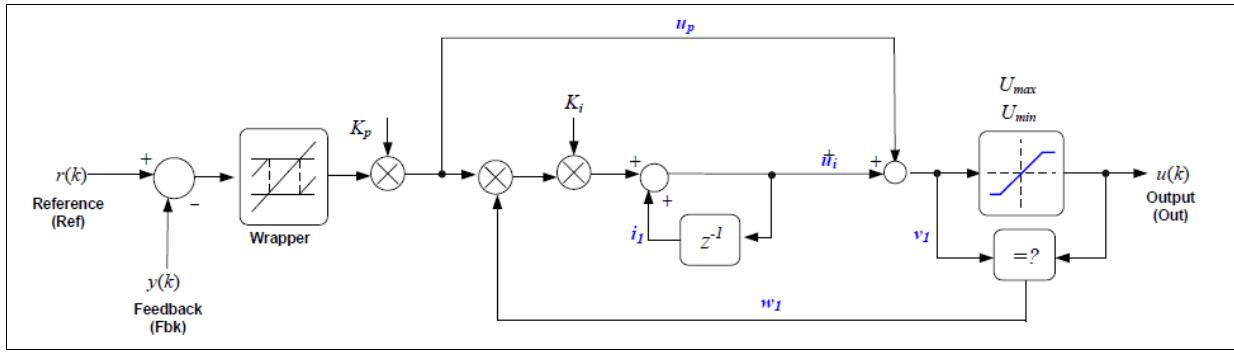
Parameters	Description
Proportional Gain Kp	Proportional loop gain
Integral Gain Ki	Integral gain
Maximum Output	Maximum output limit
Minimum Output	Minimum output limit

13.12.3 PI_POS: PI Controller with Position Error Wrapper

This block implements a simple 32-bit digital PI controller with anti-windup correction, and also with a position error wrapper. It consists of a basic summing junction and P+I control law with the following features:

- Programmable output saturation
- Independent reference weighting on proportional path
- Anti-windup integrator reset.
- Position error wrap around to fit within $-\pi$ and $+\pi$.

The PI_POS controller is a sub-set of the PID controller. All input, output and internal data are in IQ24 fixed-point format. The block diagram of the internal controller structure is shown below.



Attributes:

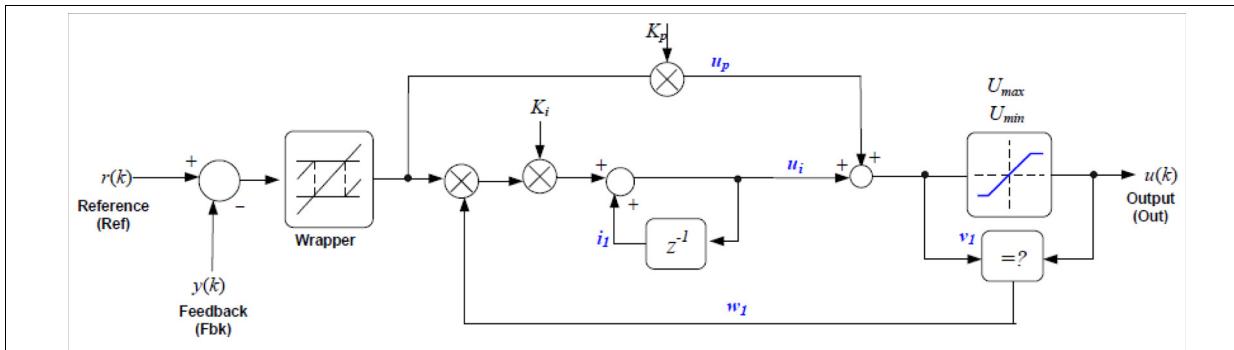
Parameters	Description
Proportional Gain Kp	Proportional loop gain
Integral Gain Ki	Integral gain
Maximum Output	Maximum output limit
Minimum Output	Minimum output limit

13.12.4 PI_POS_REG4: PI Controller with Position Error Wrapper

This module implements a generic, simple 32-bit digital PI controller with anti-windup correction, exactly same as in the previous section on PI_POS controller but treated as in PI_REG4. This block implements a simple 32-bit digital PI controller with anti-windup correction, and also with a position error wrapper. It consists of a basic summing junction and P+I control law with the following features:

- Programmable output saturation
- Independent reference weighting on proportional path
- Anti-windup integrator reset.
- Position error wrap around to fit within $-\pi$ and $+\pi$.

The PI_POS_REG4 controller is a sub-set of the PID controller. All input, output and internal data are in IQ24 fixed-point format. The block diagram of the internal controller structure is shown below.



Attributes:

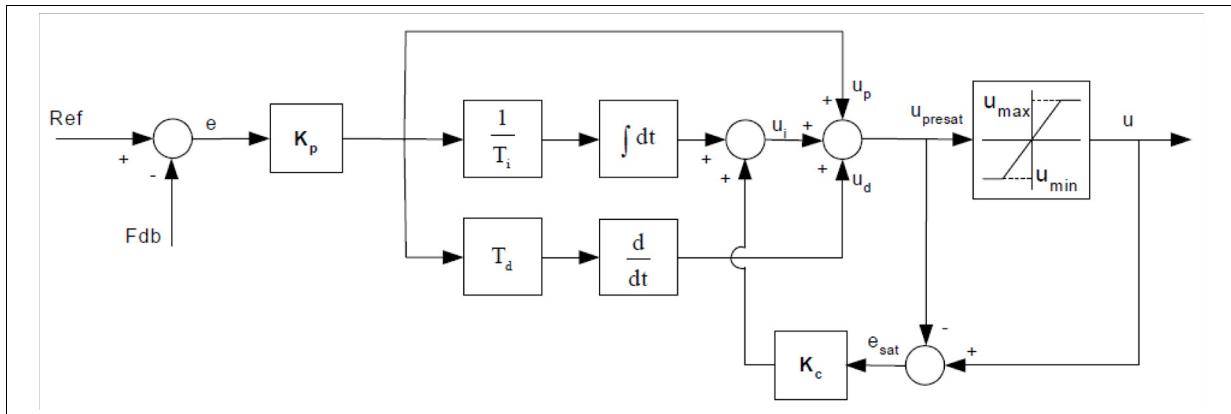
Parameters	Description
Proportional Gain Kp	Proportional loop gain
Integral Gain Ki	Integral gain

Maximum Output	Maximum output limit
Minimum Output	Minimum output limit

13.12.5 PID_REG3: PID Controller with Anti-Windup

This block implements a 32-bit digital PID controller with anti-windup correction. It can be used for PI or PD controller as well. In this digital PID controller, the differential equation is transformed to the difference equation by means of the backward approximation.

The block diagram of this conventional PID controller with anti-windup correction is shown below.



Attributes:

Parameters	Description
Proportional Loop Gain Kp	Proportional loop gain
Integral Gain Ki	Integral gain
Derivative Gain Kd	Derivative gain
Integral Correction Gain Kc	Integral correction gain
Maximum Output	Maximum output limit
Minimum Output	Minimum output limit

13.12.6 PID_GRANDO: PID Controller

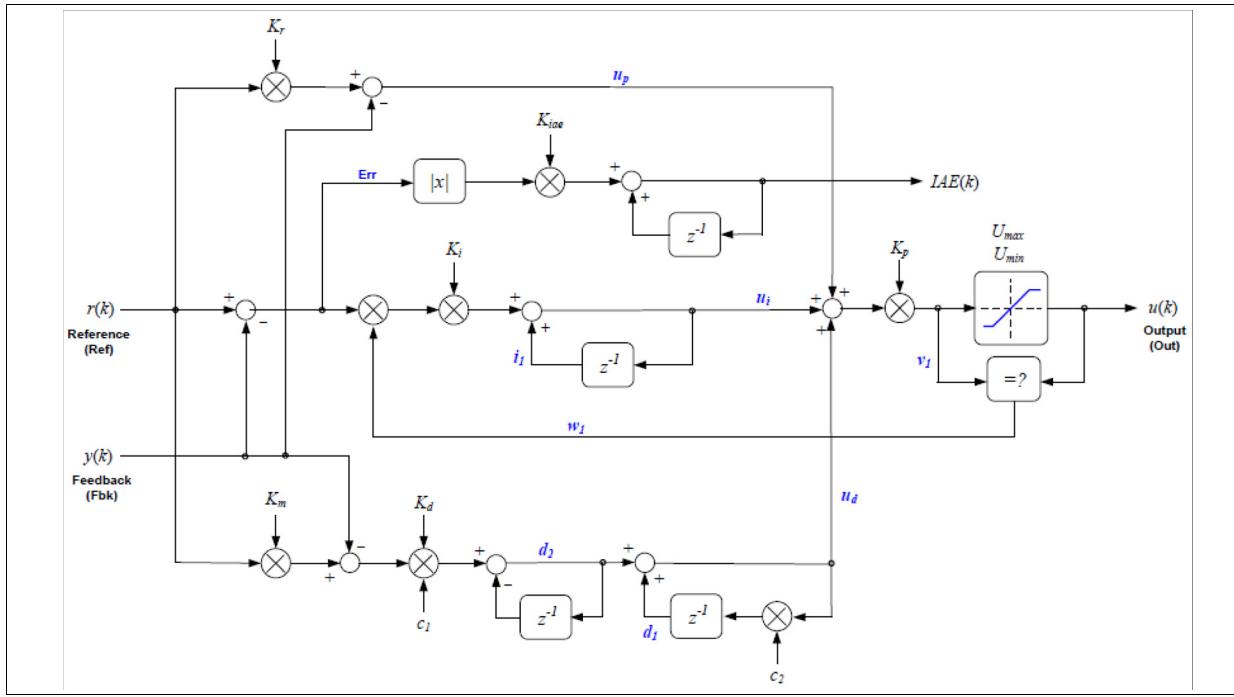
This block implements a basic summing junction and PID control law with the following features:

- Programmable output saturation
- Independent reference weighting on proportional path
- Independent reference weighting on derivative path
- Anti-windup integrator reset.
- Programmable derivative filter
- Transient performance measurement

PID Grando is an example of a PID structure often called "standard" form, in which proportional gain is applied after the three controller paths have been summed. This contrasts with the "parallel" PID form, in which P, I, and D gains are applied in separate paths. All input, output and internal data is in IQ24 fixed-point format.

The Grando controller includes a saturation block to limit the range of the control effort, $u(k)$. If the output saturates, the integrator is disabled to prevent a phenomenon known as "wind-up". In cases where saturation may occur in other parts of the control loop, user code should disable integral action by temporarily setting the integrator gain (Ki) to zero when saturation occurs, and restoring it once saturation has been cleared.

The block diagram of the internal controller structure is shown below.



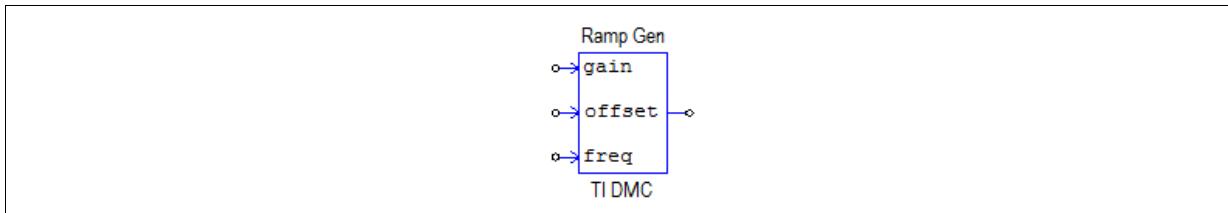
Attributes:

Parameters	Description
Proportional Ref Weight Kr	Reference weighting on proportional path
Proportional Loop Gain Kp	Proportional loop gain
Integral Gain Ki	Integral gain
Derivative Gain Kd	Derivative gain
Derivative Ref. Weight Km	Derivative reference weighting
Cut-off Frequency fc	Cut-off frequency for the first order filter on derivative path
Maximum Output Umax	Maximum output limit
Minimum Output Umin	Minimum output limit
Sampling Frequency	System sampling frequency, in Hz

13.13 RAMP_GEN: Ramp Generator

This block generates ramp output of adjustable gain, frequency, and dc offset.

Image:



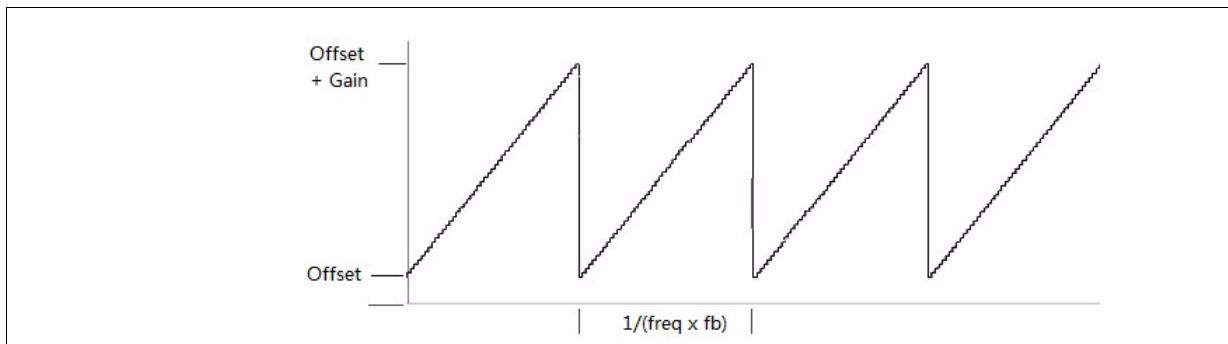
Attributes:

Parameters	Description
Base Frequency fb	Base frequency, in Hz
Sampling Frequency	Sampling frequency, in Hz

Input and Output Signals:

Name	I/O	Description
gain	Input	Ramp gain, in the range of 0 to 2.
Offset	Input	Ramp offset, in the range of -1 to +1. The value of offset+gain must be less than 1.0.
freq	Input	Ramp frequency ratio. The ramp output frequency is freq*fb
	Output	Ramp signal output

The ramp output waveform is shown below.



The output ramp frequency is the multiple of the base frequency.

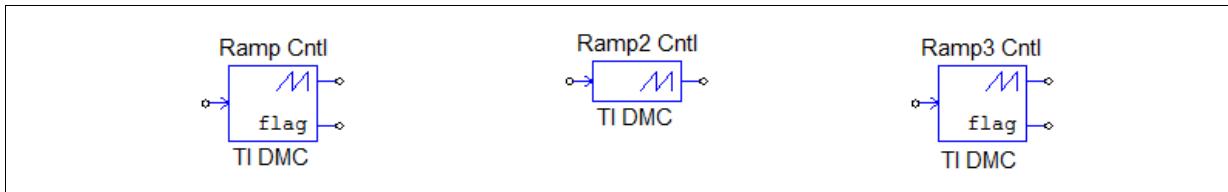
The output range is limited from -1 to +1. Therefore, the range of *Offset* is from -1 to +1 and the range of *Gain* is from 0 to 2. The value of *Offset+Gain* must be less than 1.0.

13.14 Ramp Control

There are three different ramp control blocks corresponding to the three ramp macros in the TI DMC library:

- RMP_CNTL: Implements a ramp up and down, with an output to flag when output equals input.
- RMP2CNTL: Implements a ramp up and down.
- RMP3CNTL: Implements a ramp down, with an output to flag when output equals input.

Images:



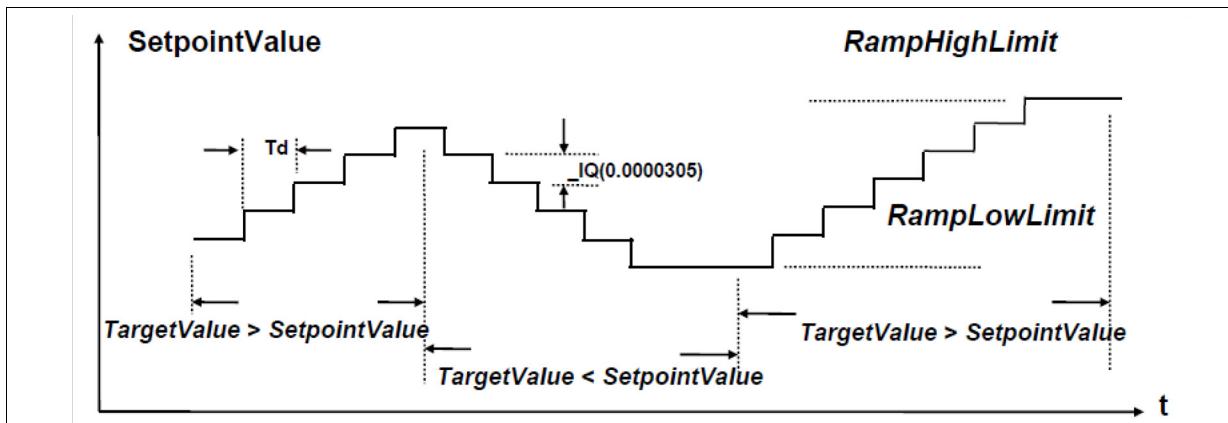
Input and Output Signals:

Name	I/O	Description
flag	Input	Target input value
	Output	Target output
	Output	Flag to indicate output equals input (not available for RMP2CNTL).

13.14.1 RMP_CNTL: Ramp Control

This block implements a ramp up and ramp down function with a flag to indicate when the output variable equals the input variable

- The ramp step = IQ(0.0000305).
- For input > output: output increments one ramp step after delay if it is below the Maximum Limit.
- For input < output: output decrements one ramp step after delay if it is above the Minimum Limit.
- For input = output: the flag is set to 7FFFFFFFh.



Example:

SetpointValue = 0 (initial value), TargetValue = 1000 (user specified), RampDelayMax = 500 (user specified), and sampling loop time period Ts = 0.000025 sec.

This means that the time delay for each ramp step is $Td = 500 \times 0.000025 = 0.0125$ sec. Therefore, the total ramp time will be $Tramp = 1000 \times 0.0125 = 12.5$ sec.

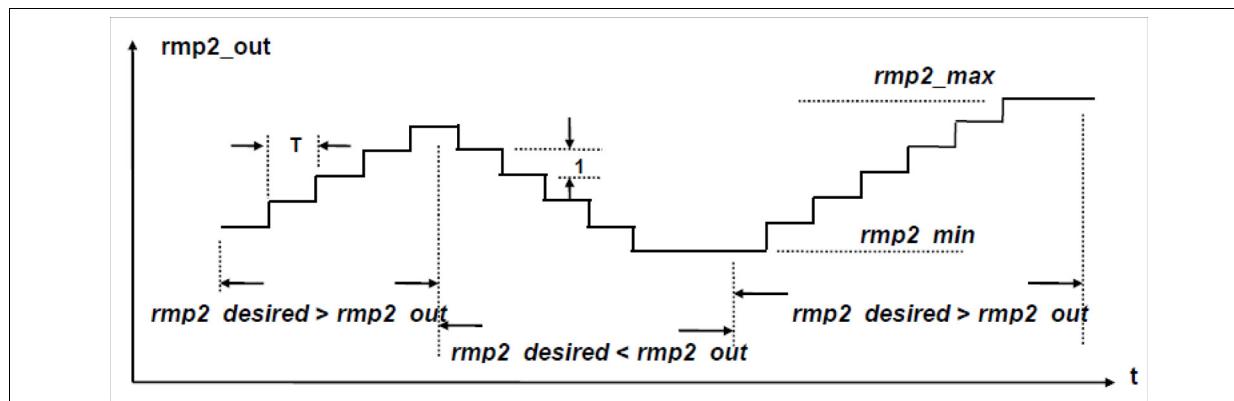
Attributes:

Parameters	Description
Max Delay Rate	Delay rate for each ramp step, in number of sampling time period
Minimum Output	Minimum output limit
Maximum Output	Maximum output limit

13.14.2 RMP2CNTL: Ramp 2 Control

This block implements a ramp up and ramp down function. The output follows the input.

- The ramp step = 1.
- If input > output: output increments one ramp step after delay if it is below the maximum output.
- If input < output: output decrements one ramp step after delay if it is above the minimum output.



Example:

Out = 0 (initial value), DesiredInput = 1000 (user specified), Ramp2Delay = 500 (user specified), and sampling loop time period Ts = 0.000025 sec.

This means that the time delay for each ramp step is $T_d = 500 \times 0.000025 = 0.0125$ sec. Therefore, the total ramp time will be $T_{ramp} = 1000 \times 0.0125 = 12.5$ sec.

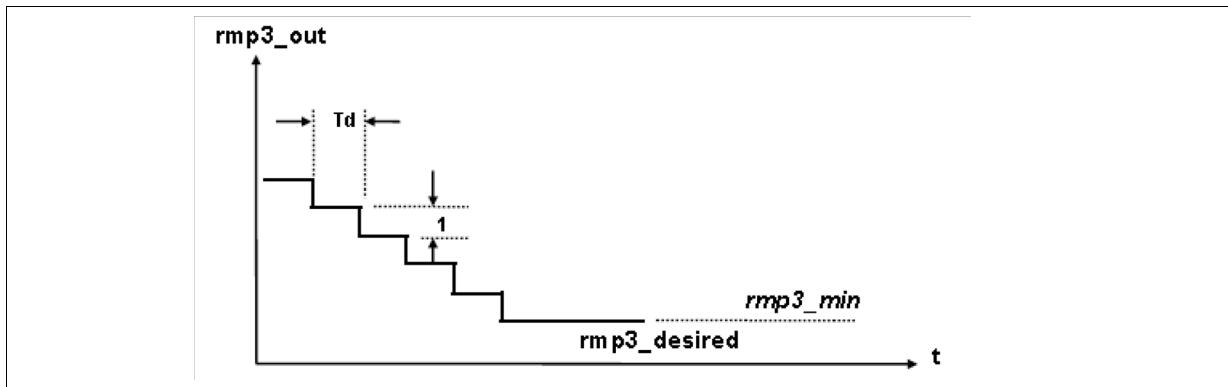
Attributes:

Parameters	Description
Delay in # of Period	The delay rate for each ramp step, in number of sampling time period.
Minimum Output	Minimum output limit
Maximum Output	Maximum output limit
Initial Output Value	Initial output value

13.14.3 RMP3CNTL: Ramp 3 Control

This block implements a ramp down function.

- The ramp step = 1.
- Output decrements one ramp step after delay if it is above the Minimum Limit.
- If output = input: the flag is set to 7FFFh.



Example:

Out=500(initial value), DesiredInput=20(user specified),

Ramp3Delay=100(user specified), sampling loop time period Ts=0.000025 Sec.

This means that the time delay for each ramp step is $Td=100 \times 0.000025=0.0025$ Sec. Therefore, the total ramp down time will be $T_{ramp}=(500-20) \times 0.0025$ Sec=1.2 Sec

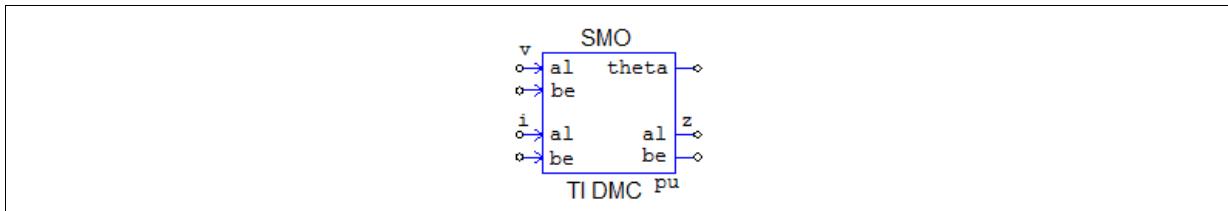
Attributes:

Parameters	Description
Delay in # of Period	The delay rate for each ramp step, in number of sampling time period.
Maximum Output	Maximum output limit
Minimum Output	Minimum output limit
Initial Output Value	Initial output value

13.15 SOMPOS: Sliding-Mode Rotor Position Observer

This block implements a rotor position estimation algorithm for permanent-magnet synchronous motor (PMSM) based on sliding-mod observer (SMO).

Image:



Attributes:

Parameters	Description
Sliding-Mode Control Gain	Sliding-mode control gain (Kslide).
Sliding-Mode Filter Gain	Sliding -mode filter gain (Kslf).

Stator Resistance Rs	Stator resistance, in ohm.
Stator inductance Ls	Stator inductance, in H.
Base Phase Current Ib	Base phase current, in Amp.
Base Phase Voltage Vb	Base phase voltage, in volt.
Sampling Frequency	System sampling frequency, in Hz

Input and Output Signals:

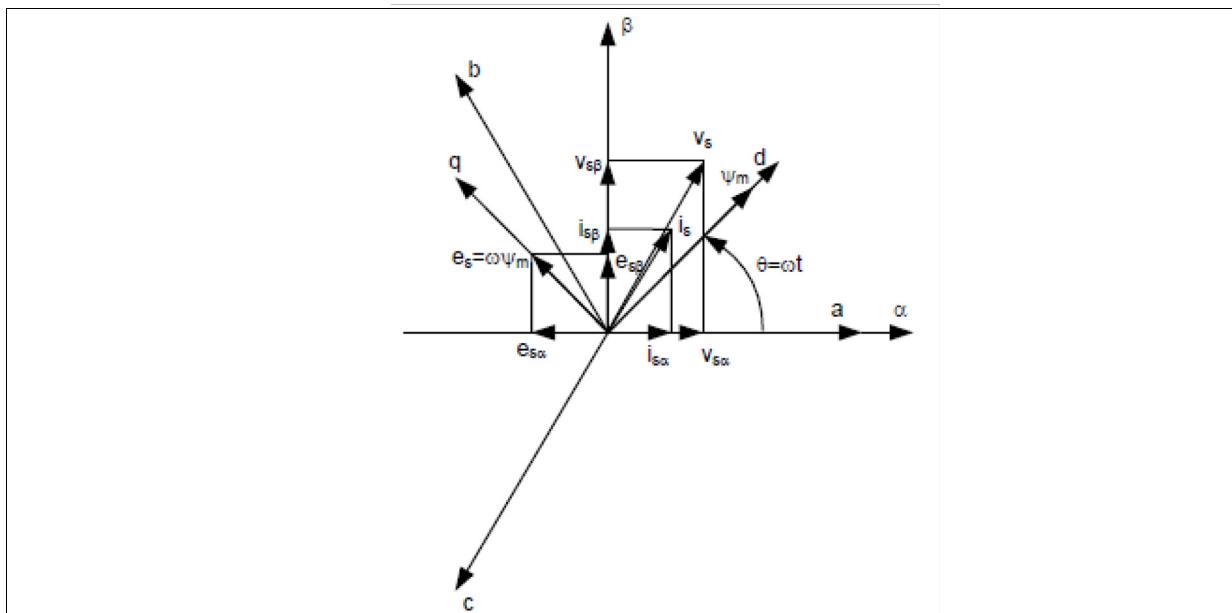
All input and output signals are in per unit.

Name	I/O	Description
v-al	Input	Stationary alpha-axis stator voltage
v-be	Input	Stationary beta-axis stator voltage
i-al	Input	Stationary alpha-axis stator current
i-be	Input	Stationary beta-axis stator current
theta	Output	Compensated rotor position angle
z-al	Output	Stationary alpha-axis stator sliding control
z-be	Output	Stationary beta-axis stator sliding control

The sliding-mode rotor position observer of PMSM module requires two constants (Fsmopos and Gsmopos) to be input basing on the machine parameters, base quantities, mechanical parameters, and sampling period. These two constants are computed by the macro SMOPOS_CONST.

SimCoder combines SMOPOS and SMOPOS_CONST in this block to simplify the process. When a SMOPOS element is in a circuit schematic, SimCoder copies SMOPOS macro to the generated project folder and implements SMOPOS_CONST sub-module in SMOPOS's initial data structure definition.

The figure below is an illustration of the coordinate frames and voltage and current vectors of PMSM, with a , b and c being the phase axes, α and β being a fixed Cartesian coordinate frame aligned with phase a , and d and q being a rotating Cartesian coordinate frame aligned with rotor flux. v_s , i_s and e_s are the motor phase voltage, current and back emf vectors (each with two coordinate entries). All vectors are expressed in α - β coordinate frame for the purpose of this discussion. The α - β frame expressions are obtained by applying Clarke transformation to their corresponding three phase representations.

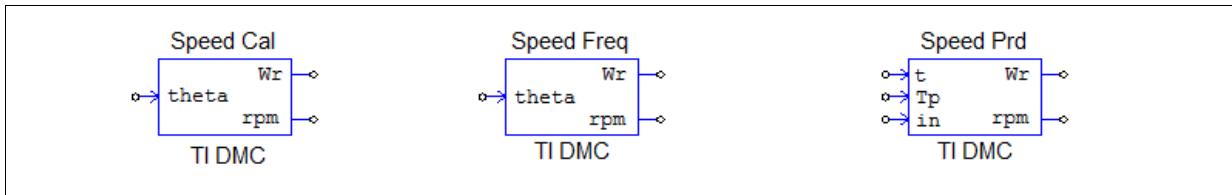


13.16 Speed Calculators

SimCoder built three different Speed Calculator blocks corresponding to the three macros in the TI DMC library:

- SPEED_EST: calculates the motor speed based on the estimated rotor position when the rotation direction information is not available.
- SPEED_FR: calculates the motor speed based on a rotor position measurement from QEP sensor.
- SPEED_PRD: calculates the motor speed based on a signal's period measurement.

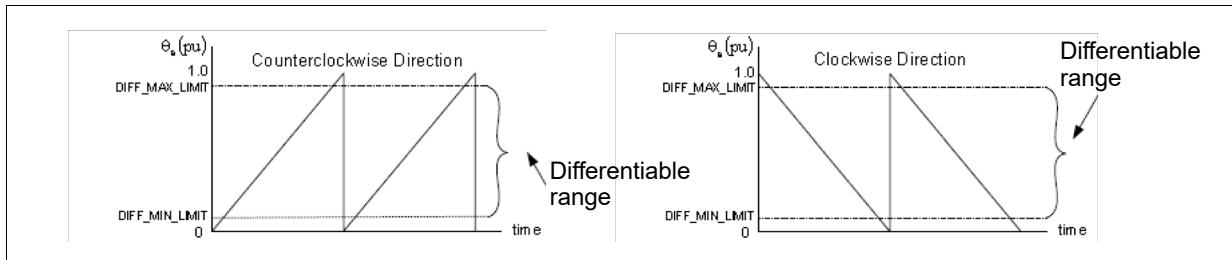
Images:



13.16.1 SPEED_EST: Speed Calculator

This block calculates the motor speed based on the estimated rotor position angle when the rotation direction information is not available. A first-order low-pass filter is used at the output variable.

The typical waveforms of the electrical rotor position angle θ in both directions are shown below. Assuming the direction of rotation is not available. To take care the discontinuity of angle from 360 to 0 degree (CCW) or from 0 to 360 degree (CW), the differentiator is simply operated only within the differentiable range as seen in this Figure. This differentiable range does not significantly lose the information to compute the estimated speed. In the figure below, $\theta = 1.0 \text{ pu} = 360 \text{ degree}$.



Attributes:

Parameters	Description
Filter Cut-off Frequency fc	The cut-off frequency of the low-pass filter at the output variable, in Hz
Base Frequency fb	Base frequency, in Hz
Base Speed (rpm)	Base speed, in rpm
Sampling Frequency	System sampling frequency, in Hz

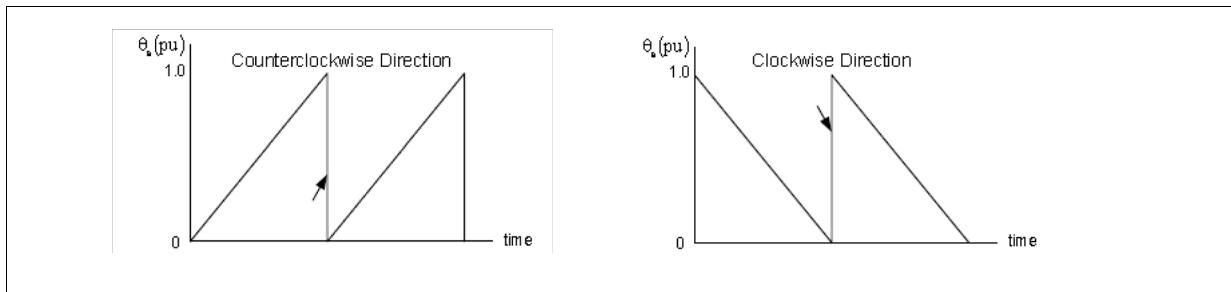
Input and Output Signals:

Name	I/O	Description
theta	Input	Estimated rotor position angle, in degrees from 0 to 360.
Wr	Output	Estimated motor speed, in per unit.
rpm	Output	Estimated motor speed, in rpm.

13.16.2 SPEED_FR: Speed Calculator with QEP Sensor

This block calculates the motor speed based on the rotor position measurement from QEP sensor. A first-order low-pass filter is used at the output variable.

The typical waveforms of the electrical rotor position angle θ in both directions can be shown as in the figure below. Assuming the direction of rotation is not available, speed is estimated based on differentiation of angular values between successive iterations. To take care of the discontinuity of angle from 360 to 0 degrees (CCW) or from 0 to 360 degrees (CW), an error roll over to fit the difference numerically within -180 and +180 degrees is performed.



Attributes:

Parameters	Description
Filter Cut-off Frequency f_c	The cut-off frequency of the low-pass filter at the output variable, in Hz
Base Frequency f_b	Base frequency, in Hz
Base Speed (rpm)	Base speed, in rpm
Sampling Frequency	System sampling frequency, in Hz

Input and Output Signals:

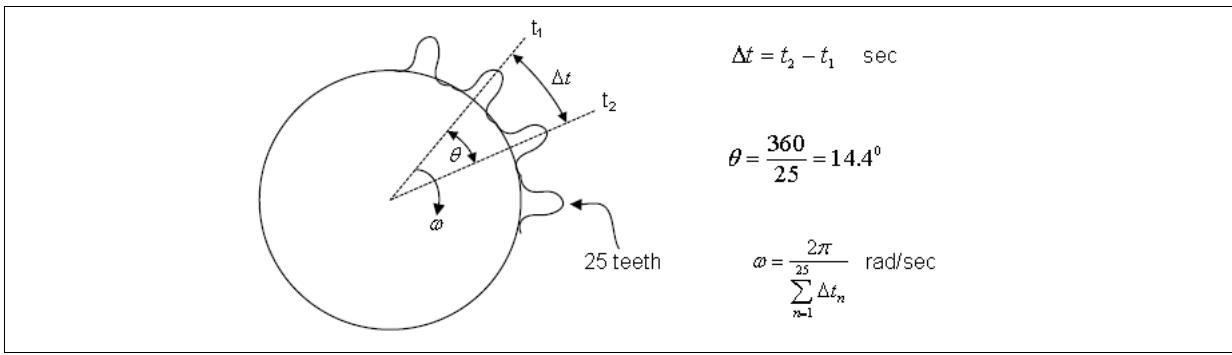
Name	I/O	Description
theta	Input	Estimated rotor position angle, in degrees from 0 to 360.
Wr	Output	Estimated motor speed, in per unit.
rpm	Output	Estimated motor speed, in rpm.

13.16.3 SPEED_PRD: Speed Calculator with Period Measurement

This block calculates the motor speed based on the measurement of a signal's period. Such a signal, for which the period is measured, can be the periodic output pulses from a motor speed sensor such as a shaft sprocket together with a Hall effect gear tooth sensor.

A low cost shaft sprocket with n teeth and a Hall effect gear tooth sensor is used to measure the motor speed. The figure below shows the physical details associated with the sprocket. The Hall effect sensor outputs a square wave pulse every time a tooth rotates within its proximity. The resultant pulse rate is n pulses per revolution. The Hall effect sensor output is fed directly to the Capture input pin. The capture unit will capture (the value of its base timer counter) on either the rising or the falling edges (whichever is specified) of the Hall effect sensor output. The captured value is passed to this s/w module through the variable called TimeStamp.

In this module, every time a new input TimeStamp becomes available it is compared with the previous TimeStamp. Thus, the tooth-to-tooth period ($t_2 - t_1$) value is calculated. In order to reduce jitter or period fluctuation, an average of the most recent n period measurements can be performed each time a new pulse is detected.



Attributes:

Parameters	Description
# of Sprocket Teeth	The number of sprocket teeth, an integer number
Base Speed (rpm)	Base speed, in rpm
Sampling Frequency	System sampling frequency, in Hz

Input and Output Signals:

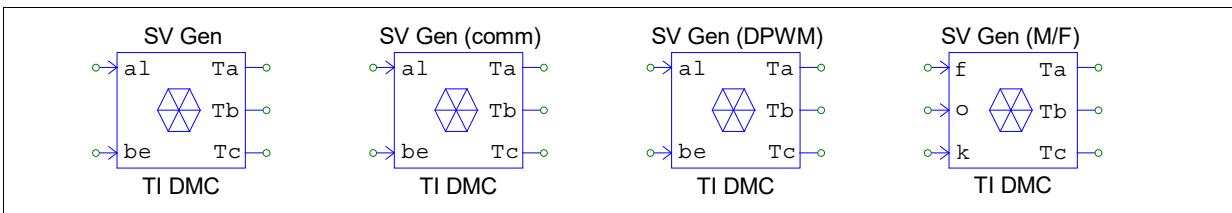
Name	I/O	Description
t	Input	Time stamp for a capture event.
Tp	Input	Event period between time stamps.
in	Input	Input selection. 0 for time stamp; 1 for event period
Wr	Output	Estimated motor speed, in per unit.
rpm	Output	Estimated motor speed, in rpm.

13.17 Space Vector Generators

Space vector generators calculates the appropriate duty ratios needed to generate a given stator voltage using space vector PWM technique. SimCoder built different space vector blocks corresponding to the two macros in the TI DMC library:

- **SVGEN**: the stator reference DPW voltage is calculated from its alpha-beta components.
- **SVGEN_COMM**: The stator reference voltage is calculated using common mode voltage.
- **SVGEN_PWM**: Different than regular SVGEN, this modulation technique keeps one of the three switches off during the entire 120 degrees to minimize switching losses.
- **SVGEN_MF**: The stator reference voltage is calculated from its magnitude and frequency.

Images:



Input and Output Signals:

Name	I/O	Description
al	Input	Reference alpha-axis phase voltage.
be	Input	Reference beta-axis phase voltage.
Ta	Output	Duty ratio reference for phase-a switching function.
Tb	Output	Duty ratio reference for phase-b switching function.
Tc	Output	Duty ratio reference for phase-c switching function.

For Blocks SVGEN, SVGEN_COMM, and SVGEN_DPWM: There are no parameters for these blocks.

For Block SVGEN_MF:

Attributes:

Parameters	Description
Base Frequency	Base frequency, in Hz
Sampling Frequency	System sampling frequency, in Hz

Input and Output Signals:

Name	I/O	Description
f	Input	Reference frequency, in Hz
o	Input	Reference offset voltage, in volt
k	Input	Reference gain voltage
Ta	Output	Duty ratio reference for phase-a switching function.
Tb	Output	Duty ratio reference for phase-b switching function.
Tc	Output	Duty ratio reference for phase-v switching function.

13.18 VHZ_PROFILE: Volt/Hertz Profile for AC Induction Motors

This block generates an output voltage for a specific input frequency according to the specific volt/hertz profile. This is used for variable speed implementation of AC induction motor drives.

The relationship between the output voltage and the input frequency is illustrated below.

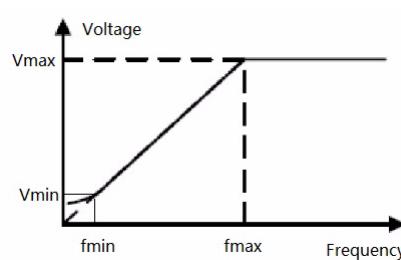
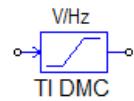


Image:**Attributes:**

Parameters	Description
Low Frequency (pu)	Low frequency f_{min} , in per unit
High Frequency (pu)	High frequency f_{rated} at the rated voltage V_{rated} , in per unit
Maximum Frequency (pu)	Maximum frequency, in per unit
Rated Voltage (pu)	Rated voltage V_{rated} , in per unit
Low Freq. Voltage (pu)	Voltage at the low frequency f_{min} , in per unit

Input and Output Signals:

Name	I/O	Description
	Input Output	Frequency Voltage